Solace JMS Integration with Red Hat JBoss Fuse (6.0)

Document Version 1.1

November 2015

This document is an integration guide for using Solace JMS as a JMS provider in Red Hat JBoss Fuse.

Red Hat JBoss Fuse product is an open source Enterprise Service Bus (ESB). It delivers a robust, cost-effective, and open integration platform that lets enterprises easily connect their disparate applications, services, or devices in real time. An integrated enterprise is able to provide better products and services to its customers. A flexible architecture coupled with popular and proven integration tools enables Red Hat JBoss Fuse to provide integration everywhere.

The Solace message router supports persistent and non-persistent JMS messaging with high throughput and low, consistent latency. Thanks to very high capacity and built-in virtualization, each Solace message router can replace dozens of software-based JMS brokers in multi-tenant deployments. Since JMS is a standard API, client applications connect to Solace like any other JMS broker so companies whose applications are struggling with performance or reliability issues can easily overcome them by upgrading to Solace's hardware.

Solace Systems®

http://www.solacesystems.com

Table of Contents

1 Overview

This document demonstrates how to integrate Solace Java Message Service (JMS) with Red Hat JBoss Fuse v6.0 for production and consumption of JMS messages. The goal of this document is to outline best practices for this integration to enable efficient use of both JBoss Fuse and Solace JMS.

The target audience of this document is developers using JBoss Fuse with knowledge of both JBoss Fuse and JMS in general. As such this document focuses on the technical steps required to achieve the integration. For detailed background on either Solace JMS or JBoss Fuse refer to the referenced documents below.

This document is divided into the following sections to cover the Solace JMS integration with JBoss Fuse:

- Integrating with JBoss Fuse
- Performance Considerations
- Working with Solace High Availability

1.1 Related Documentation

These documents contain information related to the feature defined in this document

Document ID	Document Title	Document Source
[Solace-Portal]	Solace Developer Portal	http://dev.solacesystems.com
[Solace-JMS-REF]	Solace Messaging API for JMS Developer Guide	http://dev.solacesystems.com/docs/solace-jms-api-developer-guide
[Solace-JMS-API]	Solace JMS API Online Reference Documentation	http://dev.solacesystems.com/docs/solace-jms-api-online-reference
[Solace-FG]	Solace Messaging Platform – Feature Guide	http://dev.solacesystems.com/docs/messaging-platform-feature-guide
[Solace-FP]	Solace Messaging Platform – Feature Provisioning	http://dev.solacesystems.com/docs/messaging-platform-feature-provisioning
[Solace-CLI]	Solace Appliance Command Line Interface Reference	http://dev.solacesystems.com/docs/cli-reference
[JBoss Fuse – REF]	RedHat JBoss_Fuse Online reference documentation	https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/
[JBoss Fuse – JMS]	Using the JMS Binding Component	https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/6.0/html/Using_the_JMS_Binding_Component/index.html

Table 1 - Related Documents

2 Why Solace

Solace technology efficiently moves information between all kinds of applications, users and devices, anywhere in the world, over all kinds of networks. Solace makes its state-of-the-art data movement capabilities available via hardware and software “message routers” that can meet the needs of any application or deployment environment. Solace’s unique solution offers unmatched capacity, performance, robustness and TCO so our customers can focus on seizing business opportunities instead of building and maintaining complex data distribution infrastructure.

Superior Performance

Solace’s hardware and software messaging middleware products can cost-effectively meet the performance needs of any application, with feature parity and interoperability that lets companies start small and scale to support higher volume or more demanding requirements over time, and purpose-built appliances that offer 50-100x higher performance than any other technology for customers or applications that require extremely high capacity or low latency.

Robustness

Solace offers high availability (HA) and disaster recovery (DR) without the need for 3rd party products, and fast failover times no other solution can match. Distributing data via dedicated TCP connections ensures an orderly, well-behaved system under load, and patented techniques ensure that the performance of publishers and high-speed consumers is never impacted by slow consumers.

Simple Architecture

Modern enterprises run applications that demand many kinds of data movement such as persistent messaging, web streaming, WAN distribution and cloud-based communications. By supporting all kinds of data movement with a unified platform that can be deployed as a small-footprint software broker or high-capacity rack-mounted appliance, Solace lets architects design an end-to-end infrastructure that’s easy to build applications for, integrate with existing technologies, secure and scale.

Simple Operations

Solace’s solution features a shared administration framework for all kinds of data movement, deployment models and network environments so it’s easy for IT staff to deploy, monitor, manage and upgrade their Solace-based messaging environment.

Cost Savings

Solace reduces expenses with high-capacity hardware, flexible software, and the ability to deploy the right solution for each problem. Solace’s support for many kinds of messaging lets you replace multiple messaging products with just one, built-in HA, DR, WAN and Web functionality eliminate the need for third-party products.

3 Integrating with JBoss Fuse

This integration guide demonstrates how to configure JBoss Fuse to send and receive JMS messages using a shared JMS connection. Accomplishing this requires completion of the following steps.

- Step 1 – Configuration of the Solace Appliance.
- Step 2 – Configuring JBoss Fuse to connect to the Solace appliance.
- Step 3 – Configuring JBoss Fuse to send messages using Solace JMS.
- Step 4 – Configuring JBoss Fuse to receive messages using Solace JMS.

3.1 Description of Resources Required

This integration guide will demonstrate creation of Solace resources and configuration of JBoss Fuse managed resources. This section outlines the resources that are created and used in the subsequent sections.

3.1.1 Solace Resources

The following Solace appliance resources are required.

Resource	Value	Description
Solace appliance IP:Port	__IP:Port__	The IP address and port of the Solace appliance message backbone. This is the address client's use when connecting to the Solace appliance to send and receive message. This document uses a value of __IP:PORT__.
Message VPN	Solace_Fuse_VPN	A Message VPN, or virtual message broker, to scope the integration on the Solace appliance.
Client Username	fuse_user	The client username.
Client Password	fuse_password	Optional client password.
Solace Queue	Q/requests	Solace destination of messages produced and consumed
JNDI Connection Factory	JNDI/CF/fuse	The JNDI Connection factory for controlling Solace JMS connection properties
JNDI Queue Name	JNDI/Q/requests	The JNDI name of the queue used in the samples

Table 2 – Solace Configuration Resources

3.1.2 JBoss Fuse Configuration Resources

Resource	Value
JndiTemplate	Solace.JndiTemplate
JndiObjectFactoryBean	Solace.JndiObjectFactoryBean
JndiDestinationResolver	Solace.JndiDestinationResolver
JmsComponent	Solace.JmsComponent

3.2 Step 1 – Configuring the Solace Appliance

The Solace appliance needs to be configured with the following configuration objects at a minimum to enable JMS to send and receive messages within JBoss Fuse.

- A Message VPN, or virtual message broker, to scope the integration on the Solace appliance.
- Client connectivity configurations like usernames and profiles
- Guaranteed messaging endpoints for receiving messages.
- Appropriate JNDI mappings enabling JMS clients to connect to the Solace appliance configuration.

For reference, the CLI commands in the following sections are from SolOS version 6.2 but will generally be forward compatible. For more details related to Solace appliance CLI see [Solace-CLI]. Wherever possible, default values will be used to minimize the required configuration. The CLI commands listed also assume that the CLI user has a Global Access Level set to Admin. For details on CLI access levels please see [Solace-FG] section “User Authentication and Authorization”.

Also note that this configuration can also be easily performed using SolAdmin, Solace’s GUI management tool. This is in fact the recommended approach for configuring a Solace appliance. This document uses CLI as the reference to remain concise.

3.2.1 Creating a Message VPN

This section outlines how to create a message-VPN called “Solace_Fuse_VPN” on the Solace appliance with authentication disabled and 2GB of message spool quota for Guaranteed Messaging. This message-VPN name is required in JBoss Fuse configuration when connecting to the Solace messaging appliance. In practice appropriate values for authentication, message spool and other message-VPN properties should be chosen depending on the end application’s use case.

```
(config)# create message-vpn Solace_Fuse_VPN
(config-msg-vpn)# authentication
(config-msg-vpn-auth)# user-class client
(config-msg-vpn-auth-user-class)# basic auth-type none
(config-msg-vpn-auth-user-class)# exit
(config-msg-vpn-auth)# exit
(config-msg-vpn)# no shutdown
(config-msg-vpn)# exit
(config)#
(config)# message-spool message-vpn Solace_Fuse_VPN
(config-message-spool)# max-spool-usage 2000
(config-message-spool)# exit
(config)#
```

3.2.2 Configuring Client Usernames & Profiles

This section outlines how to update the default client-profile and how to create a client username for connecting to the Solace appliance. For the client-profile, it is important to enable guaranteed messaging for JMS messaging and transacted sessions if using transactions.

The chosen client username of “fuse_user” will be required by JBoss Fuse when connecting to the Solace appliance.

```
(config)# client-profile default message-vpn Solace_Fuse_VPN
(config-client-profile)# message-spool allow-guaranteed-message-receive
(config-client-profile)# message-spool allow-guaranteed-message-send
(config-client-profile)# message-spool allow-transacted-sessions
(config-client-profile)# exit
(config)#
(config)# create client-username fuse_user message-vpn Solace_Fuse_VPN
(config-client-username)# acl-profile default
(config-client-username)# client-profile default
(config-client-username)# no shutdown
(config-client-username)# exit
(config)#
```

3.2.3 Setting up Guaranteed Messaging Endpoints

This integration guide shows receiving messages within JBoss Fuse from a single JMS Queue. For illustration purposes, this queue is chosen to be an exclusive queue with a message spool quota of 2GB matching quota associated with the message VPN. The queue name chosen is “Q/requests”.

```
(config)# message-spool message-vpn Solace_Fuse_VPN
(config-message-spool)# create queue Q/requests
(config-message-spool-queue)# access-type exclusive
(config-message-spool-queue)# max-spool-usage 2000
(config-message-spool-queue)# permission all delete
(config-message-spool-queue)# no shutdown
(config-message-spool-queue)# exit
(config-message-spool)# exit
(config)#
```

3.2.4 Setting up Solace JNDI References

To enable the JMS clients to connect and look up the Queue destination required by JBoss Fuse, there are two JNDI objects required on the Solace appliance:

- A connection factory: JNDI/CF/fuse
- A queue destination: JNDI/Q/requests

They are configured as follows:

```
(config)# jndi message-vpn Solace_Fuse_VPN
(config-jndi)# create connection-factory JNDI/CF/fuse
(config-jndi-connection-factory)# property-list messaging-properties
(config-jndi-connection-factory-pl)# property default-delivery-mode persistent
(config-jndi-connection-factory-pl)# exit
(config-jndi-connection-factory)# property-list transport-properties
(config-jndi-connection-factory-pl)# property direct-transport false
(config-jndi-connection-factory-pl)# property "reconnect-retry-wait" "3000"
(config-jndi-connection-factory-pl)# property "reconnect-retries" "20"
(config-jndi-connection-factory-pl)# property "connect-retries-per-host" "5"
(config-jndi-connection-factory-pl)# property "connect-retries" "1"
(config-jndi-connection-factory-pl)# exit
(config-jndi-connection-factory)# exit
(config-jndi)#
(config-jndi)# create queue JNDI/Q/requests
(config-jndi-queue)# property physical-name Q/requests
(config-jndi-queue)# exit
(config-jndi)#
(config-jndi)# no shutdown
(config-jndi)# exit
(config)#
```

3.3 Step 2 – JBoss Fuse – Connecting

For more details refer to the JBoss Fuse document [JBoss Fuse – JMS] and see the section titled “Configuring the Connection Factor”, sub-section “Using JNDI”. The following is a direct link to the Red Hat documentation related to JNDI setup.

https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/6.0/html/Using_the_JMS_Binding_Component/files/ESBJMSConnectFactoryJNDI.html

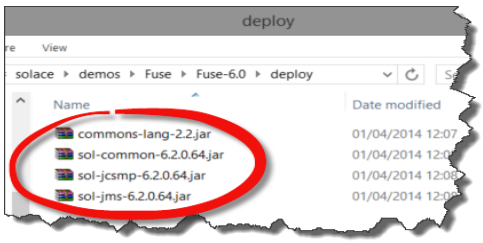
Setting up JBoss Fuse requires two steps to be completed. First, the Solace JMS libraries must be copied to JBoss Fuse. Then the correct spring configuration file must be deployed. See the following sections for details.

3.3.1 Install the Solace JMS libraries in JBoss Fuse

To install the Solace JMS libraries, copy the required Solace JMS libraries under the deploy folder of JBoss Fuse. The following is a list of required Solace JMS libraries from a 7.0 Solace JMS.

- commons-lang-<version>.jar
- sol-common-<version>.jar
- sol-jcsmp-<version>.jar
- sol-jms-<version>.jar

The following figure specifies where to deploy 3rd party (Solace) JMS libraries for JBoss Fuse to be able to pick up necessary classes when initiating connection to the Solace JMS provider.



In this case the directory where the Solace JMS libraries should be copied is %FuseHomeDirectory%\deploy.

3.3.2 JBoss Fuse Configuration

It is necessary to create and deploy a spring configuration file under deploy folder of JBoss Fuse. The following is a sample Spring configuration showing necessary configuration in order to successfully connect to Solace appliance.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:camel="http://camel.apache.org/schema/spring"
  xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">

  <bean id="Solace.JndiTemplate"
        class="org.springframework.jndi.JndiTemplate"
        lazy-init="default" autowire="default">
```

```
    <property name="environment">
      <props>
        <prop key="java.naming.provider.url">smf://__IP:PORT__</prop>
        <prop key="java.naming.factory.initial">
          com.solacesystems.jndi.SolJNDIInitialContextFactory
        </prop>
        <prop key="java.naming.security.principal">fuse_user@Solace_Fuse_VPN </prop>
      </props>
    </property>
  </bean>

  <bean id="Solace.JndiObjectFactoryBean"
        class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="jndiTemplate" ref="Solace.JndiTemplate"/>
    <property name="jndiName" value="JNDI/CF/fuse"/>
  </bean>

  <bean id="Solace.JndiDestinationResolver"
        class="org.springframework.jms.support.destination.JndiDestinationResolver">
    <property name="jndiTemplate" ref="Solace.JndiTemplate" />
    <property name="cache" value="true" />
  </bean>

  <bean id="Solace.JmsComponent" class="org.apache.camel.component.jms.JmsComponent">
    <property name="connectionFactory" ref="Solace.JndiObjectFactoryBean"/>
    <property name="destinationResolver" ref="Solace.JndiDestinationResolver" />
  </bean>

  <camelContext xmlns="http://camel.apache.org/schema/spring">
  </camelContext>
</beans>
```

The following table explains the configuration and its purpose when connecting to the Solace appliance.

Bean Id	Description
Solace.JndiTemplate	This template outlines general connection details for reaching the Solace JNDI hosted on the Solace appliance. The Solace JNDI is used to look up parameters for client connections and for destinations.
Solace.JndiObjectFactoryBean	This references a specific connection factory within the Solace JNDI that will be used when creating new connections. The value for “jndiName” is the connection factory name as configured in the Solace JNDI. In the previous section this was configured as “JNDI/CF/spring”

Bean Id	Description
Solace.JndiDestinationResolver	The `JndiDestinationResolver` allows destinations to be resolved dynamically using JNDI. Because a JNDI lookup is an expensive request, the `JndiDestinationResolver` also allows for caching of destinations through the `setCache()` method. When using this destination resolver with Solace JMS, it is very important to enable destination caching for Solace JMS to work effectively. By default this is enabled in the Spring Framework.
Solace.JmsComponent	The cached connection factory allows for re-use of the Solace connection when sending messages. For efficient integration within the Spring Framework, it is essential that connection caching be enabled and configured correctly. There are more details on this in Section 4 Performance Considerations including discussion of the `sessionCacheSize` attribute. It is this connection factory that is used by the producer and consumer clients when connecting.
camelContext	This will be used in subsequent sections to send and receive messages.

Table 3 - Solace Connection Configuration

3.4 Step 3 – JBoss Fuse – Sending Messages to Solace

To send messages to Solace, a camel route must be defined. The following modifications to the configuration file created above in Step 2.3 show an example of simple message sending.

To the previous configuration, add a route element with “to” child which specifies the Solace JMS JNDI name of the destination used for sending.

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route id="pubtosolace">
    <from uri="timer:simple?period=5000"/>
    <setBody>
      <simple>Solace Camel JMS Test</simple>
    </setBody>
    <to uri="Solace.JmsComponent:queue:JNDI/Q/requests"/>
  </route>
</camelContext>
```

With the above configuration in place JBoss Fuse when started will publish text messages every 5 seconds to Solace queue associated with a JNDI name “JNDI/Q/requests”. The content of the messages will be “<simple>Solace Camel JMS Test</simple>”.

3.5 Step 4 – JBoss Fuse – Receiving Messages from Solace

To receive messages from Solace, a camel route must be defined. The following modifications to the configuration file created above in Step 2.3 show an example of simple messages receiving.

To the previous configuration, add a route element with “from” child which specifies the Solace JMS JNDI name of the queue destination acting as source for the messages.

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route id="recvFromSolace">
    <from uri="Solace.JmsComponent:queue:JNDI/Q/requests"/>
    <to uri="log:SOLACE"/>
  </route>
</camelContext>
```

With the above configuration in place JBoss Fuse when started will subscribe to Solace queue associated with a JNDI name “JNDI/Q/requests”. Every message it consumes will be logged to the JBoss Fuse log file with a “SOLACE” tag.

4 Performance Considerations

3.1 Caching JMS Connections

In JBoss Fuse Spring, the connection object caching is controlled by the CachingConnectionFactory. A CachingConnectionFactory contains a single JMS Connection which is reused across all JmsTemplates. In order to enable session caching within the JMS Connection, the sessionCacheSize parameter must be set to specify the number of JMS Session objects to cache for reuse.

One behavior worth noting is that as outlined in the JBoss Fuse Spring documentation, if the pool of cached sessions is fully utilized and a further request for a cached `Session` is made, then the requested `Session` will be created and disposed on demand. If you couple this behavior with the fact that the default value is 1 for `sessionCacheSize` it is important to configure this to a value that is applicable to the end application to avoid the undesirable behaviour of create and dispose on each call to send a message during periods of high demand. This value should be set to the maximum concurrency required by the application.

The following configuration sample illustrates how to set the `CachingConnectionFactory` cache sizes.

```
<bean id="SolaceCachedConnectionFactory"
class="org.springframework.jms.connection.CachingConnectionFactory">
  <property name="targetConnectionFactory" ref="Solace.JndiObjectFactoryBean" />
  <property name="sessionCacheSize" value="10" />
</bean>
```

3.2 Concurrent Consumers

One can specify number of concurrent threads processing exchanges to consume messages from a non-exclusive queue in a round robin fashion. This helps consume messages from a specified queue faster. Set the JMS query option, `concurrentConsumers`, to create a thread pool of competing consumers. For example, the following route creates a pool of three competing threads that pick messages from the specified queue:

```
<route id="pubtosolace1">
  <from uri="Solace.JmsComponent:queue:JNDI/Q/requests?concurrentConsumers=5"/>
  <to uri="Solace.JmsComponent:queue:JNDI/Q/response"/>
</route>
```

Ideally value of property sessionCacheSize should be kept equals to the number of concurrent consumers to get the best performance.

5 Working with Solace High Availability (HA)

The [Solace-JMS-REF] section “Establishing Connection and Creating Sessions” provides details on how to enable the Solace JMS connection to automatically reconnect to the standby appliance in the case of a HA failover of a Solace appliance. By default Solace JMS connections will reconnect to the standby appliance in the case of an HA failover.

In general the Solace documentation contains the following note regarding reconnection:

> Note: When using HA redundant appliances, a fail-over from one appliance to its mate will typically occur in under 30 seconds, however, applications should attempt to reconnect for at least five minutes.

In section 3.2.4 Setting up Solace JNDI References, the Solace CLI commands correctly configured the required JNDI properties to reasonable values. These commands are repeated here for completeness.

```
config)# jndi message-vpn Solace_Fuse_VPN
(config-jndi)# create connection-factory JNDI/CF/fuse
(config-jndi-connection-factory)# property-list transport-properties
(config-jndi-connection-factory-pl)# property "reconnect-retry-wait" "3000"
(config-jndi-connection-factory-pl)# property "reconnect-retries" "20"
(config-jndi-connection-factory-pl)# property "connect-retries-per-host" "5"
(config-jndi-connection-factory-pl)# property "connect-retries" "1"
(config-jndi-connection-factory-pl)# exit
(config-jndi-connection-factory)# exit
(config-jndi)# exit
(config)#
```