

# Using Rhapsody 4.01 with WebLogic JMS for 10.3

Michael.W.Czapski@gmail.com

December 2011

## Table of Contents

<b>Introduction .....</b>	<b>1</b>
<b>On the WebLogic Side .....</b>	<b>1</b>
Create JMS Server.....	2
Create JMS Module .....	3
Create JMS Connection Factory .....	5
Create JMS Topic.....	6
<b>On the Rhapsody Side .....</b>	<b>8</b>
<b>Summary .....</b>	<b>11</b>

## Introduction

I had an occasion, recently, to work on an integration project which required the Rhapsody 4.01-based integration solution to receive messages from a WebLogic-based JMS Topic. Product documentation and Internet searches did not offer assistance in terms of how the Rhapsody JMS Adapter needs to be configured to support this. While there are a number of articles which discuss the topic of configuring JMS Client to interact with WebLogic JMS Server, none of the solutions described in these articles worked for me. A degree of experimentation and creative adaptation resulted in a working configuration. This article discusses this solution for the benefit of those who will be faced with this problem and for my own benefit if I need to do this again in the future.

## On the WebLogic Side

In the olden days one would need to collect a specific set of WebLogic JAR archives, needed to support JMS integration, and make them available to the client application. My article on configuring the QBrowser JMS Browser for interaction with WebLogic JMS, “Using QBrowser v2 with WebLogic JMS for 10.3”, at <http://blogs.czapski.id.au/2011/05/using-qbrowser-v2-with-weblogic-jms-for-10-3>, is an example of how this would have been done in the olden days. The method seems to have changed between then and now. The new method involves explicit creation of a client JAR, **wlfullclient.jar**, discussed amongst others in “Using the WebLogic JarBuilder Tool”, [http://docs.oracle.com/cd/E12840\\_01/wls/docs103/client/jarbuilder.html](http://docs.oracle.com/cd/E12840_01/wls/docs103/client/jarbuilder.html).

To summarise:

On WebLogic host, change directory to the WebLogic Server’s server/lib directory.

```
cd $WL_HOME/server/lib # or cd %WL_HOME%\server\lib or some such
```

Use the following command to create wlfullclient.jar in the server/lib directory:

```
java -jar wljarbuilder.jar
```

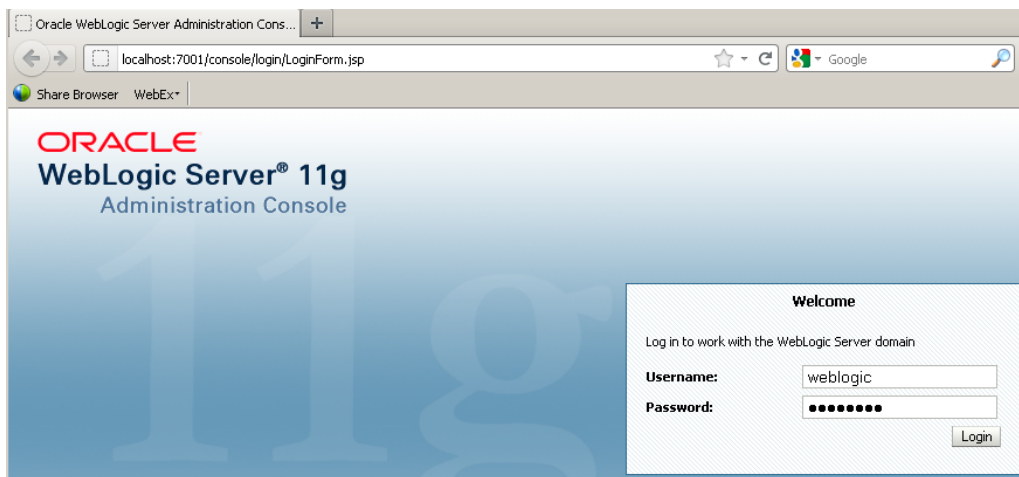
Copy the resulting `wlfullclient.jar` to the client application development area and bundle it with the client application, however these need be done for a particular client application. Add the `wlfullclient.jar` to the client application's classpath, however that needs be done for the particular client application. (Notes on how to do this for the Rhapsody JMS Adapter are provided later in this document).

The above gives you the Java Archive containing all Java classes necessary to support JMS interaction between the client application and the WebLogic Server JMS.

To make the discussion more concrete, let's assume that we have a JMS Topic Connection Factory, with a JNDI reference of "jms/MyTopicCF" and a JMS topic with a JNDI reference of "jms/MyTopic".

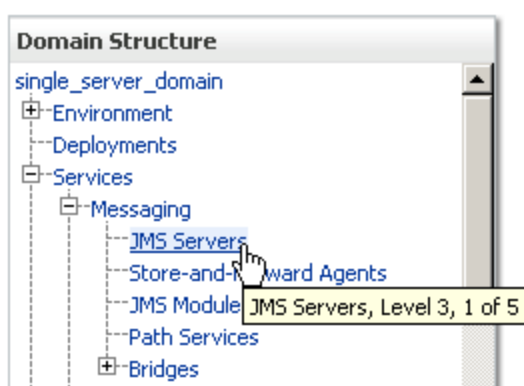
The following steps walk through the process of creating these objects using the WebLogic Admin Console.

- Log into the WebLogic Admin Console, <http://{WLJMSHost}:{WLJMSPort}>



## Create JMS Server

- Navigate the hierarchy "Services"→"Messaging" and click the "JMS Server" node



- Click "New" (This is not really necessary if there is an existing JMS Server which can be used. I don't know what your environment looks like so I am assuming you don't have a convenient JSM Serve to use)

**Summary of JMS Servers**

JMS servers act as management containers for the queues and topics in JMS modules that are targeted to them.

This page summarizes the JMS servers that have been created in the current WebLogic Server domain.

[Customize this table](#)

**JMS Servers (Filtered - More Columns Exist)**

New Delete Showing 1 to 7 of 7 Previous | Next

<input type="checkbox"/>	Name	Persistent Store	Target	Current Server	Health
<input type="checkbox"/>	BPMJMServer	BPMJMSFileStore	AdminServer	AdminServer	✔ OK
<input type="checkbox"/>	JRFWSAsyncJmsServer	JRFWSAsyncFileStore	AdminServer	AdminServer	✔ OK

- Enter "MyJMSServer" as new server name and click "Next"

**Create a New JMS Server**

Back Next Finish Cancel

**JMS Server Properties**

The following properties will be used to identify your new JMS Server.

\* Indicates required fields

What would you like to name your new JMS Server?

**Name:**

Specify persistent store for the new JMS Server.

**Persistent Store:**  Create a New Store

- Choose the target from the target list and click "Finish"

**Create a New JMS Server**

Back Next Finish Cancel

**Select targets**

Select the server instance or migratable target on which you would like to deploy this JMS Server.

**Target:**  (none) AdminServer

Back Next Finish Cancel

## Create JMS Module

- Navigate the hierarchy "Services"→"Messaging" and click the "JMS Modules" node



- Click "New" (strictly speaking creation of a JMS Module may not be necessary if you have one, for example the "SOAJMSModule" created by the installation of the SOA Suite. I don't know what your environment has so I have to assume that it does not have a JMS Module which can be used)

Customize this table

**JMS Modules**

New Delete Showing 1 to 7 of 7 Previous | Next

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	BPMJMSModule	System
<input type="checkbox"/>	JRFWSAsyncJmsModule	System
<input type="checkbox"/>	PatientJMSModule	System
<input type="checkbox"/>	SOAJMSModule	System
<input type="checkbox"/>	UMSJMSSystemResource	System

- Name the module "MyJMSModule" and click "Next"

Home > JMS Modules

**Create JMS System Module**

Back Next Finish Cancel

The following properties will be used to identify your new module.

JMS system resources are configured and stored as modules similar to standard J2E topics, connection factories, templates, destination keys, quota, distributed queues JMS store-and-forward (SAF) parameters. You can administratively configure and r resources.

\* Indicates required fields

What would you like to name your System Module?

\* Name:

- Check the "AdminServer" target and click "Next"

### Create JMS System Module

The following properties will be used to target your new JMS system module.

Use this page to select the server or cluster on which you would like to deploy this JMS system module. You can select multiple targets later if you wish.

#### Targets :

Servers
<input checked="" type="checkbox"/> AdminServer

- Click "Finish" accepting the default configuration

### Create JMS System Module

#### Add resources to this JMS system module

Use this page to indicate whether you want to immediately add resources to this JMS system module. Resources include queues, topics, connection factories, etc.

- Would you like to add resources to this JMS system module?

## Create JMS Connection Factory

- Click the link "MyJMSModule"

### JMS Modules

Showing 1 to 8 of 8 Previous | Next

<input type="checkbox"/> Name	Type
<input type="checkbox"/> BPMJMSModule	System
<input type="checkbox"/> JRFWSAsyncJmsModule	System
<input type="checkbox"/> MyJMSModule	System
<input type="checkbox"/> PatternJMSModule	System

- Click "New"

[Customize this table](#)

#### Summary of Resources

Showing 0 to 0 of 0 Previous | Next

<input type="checkbox"/> Name	Type	JNDI Name	Subdeployment	Targets
There are no items to display				

Showing 0 to 0 of 0 Previous | Next



Settings for MyJMSModule

Configuration Subdeployments Targets Security Notes

This page displays general information about a JMS system module and its resources. It also allows you to configure new resources and access existing resources.

**Name:** MyJMSModule The name of this JMS system module. [More Info...](#)

**Descriptor File Name:** jms/myjmsmodule-jms.xml The name of the JMS module descriptor file. [More Info...](#)

This page summarizes the JMS resources that have been created for this JMS system module, including queue and topic destinations, connection factories, JMS templates, destination sort keys, destination quota, distributed destinations, foreign servers, and store-and-forward parameters.

[Customize this table](#)

**Summary of Resources**

New Delete Showing 1 to 1 of 1 Previous | Next

<input type="checkbox"/>	Name ^	Type	JNDI Name	Subdeployment	Targets
<input type="checkbox"/>	MyTopicCF	Connection Factory	mys/MyTopicCF	Default Targetting	AdminServer

- Check the "Topic" radio button and click "Next"

Create a New JMS System Module Resource

Back Next Finish Cancel

Choose the type of resource you want to create.

Use these pages to create resources in a JMS system module, such as queues, topics, templates, and

Depending on the type of resource you select, you are prompted to enter basic information for creating queues and topics, connection factories, distributed queues and topics, foreign servers, and JMS SAF selecting appropriate server targets. You can also associate targetable resources with subdeployment resources and the members to server resources.

Connection Factory

Queue

Topic

- Name the Topic "MyTopic", set the JNDI Name to "jms/MyTopic" and click "Next"

Create a New JMS System Module Resource

Back Next Finish Cancel

**JMS Destination Properties**

The following properties will be used to identify your new Topic. The current module is MyJMSModule.

\* Indicates required fields

\* Name:

JNDI Name:

- Click the "Create a New Subdeployment" button

**Create a New JMS System Module Resource**

Back Next Finish Cancel

**The following properties will be used to target your new JMS system module resource**

Use this page to select a subdeployment to assign this system module resource. A subdeployment is a mechanism for a server instance, cluster, or SAF agent. If necessary, you can create a new subdeployment by clicking the **Create a New Subdeployment** button. You can also reconfigure subdeployment targets later by using the parent module's subdeployment management page.

Select the subdeployment you want to use. If you select (none), no targeting will occur.

**Subdeployments:** (none) Create a New Subdeployment

- Enter "MyTopicSubdeployment" as subdeployment name and click "OK"

**Create a New Subdeployment**

OK Cancel

**Subdeployment Properties**

The following properties will be used to identify your new subdeployment.

**Subdeployment Name:** MyTopicSubdeployment

- Check the radio button next to the "MyJMSServer" JMS Server name and click "Finish"

**Create a New JMS System Module Resource**

Back Next Finish Cancel

**The following properties will be used to target your new JMS system module resource**

Use this page to select a subdeployment to assign this system module resource. A subdeployment is a mechanism for a server instance, cluster, or SAF agent. If necessary, you can create a new subdeployment by clicking the **Create a New Subdeployment** button. You can also reconfigure subdeployment targets later by using the parent module's subdeployment management page.

Select the subdeployment you want to use. If you select (none), no targeting will occur.

**Subdeployments:** MyTopicSubdeployment Create a New Subdeployment

What targets do you want to assign to this subdeployment?

**Targets :**

JMS Servers
<input type="radio"/> BPMJMSServer
<input type="radio"/> JRFWSAsyncJMSServer
<input checked="" type="radio"/> MyJMSServer

- Inspect the list of objects under our JMS Module then close the web browser – we are done here

## On the Rhapsody Side

If the JMS Client Application which needs to communicate with the WebLogic JMS is the Rhapsody 4.01



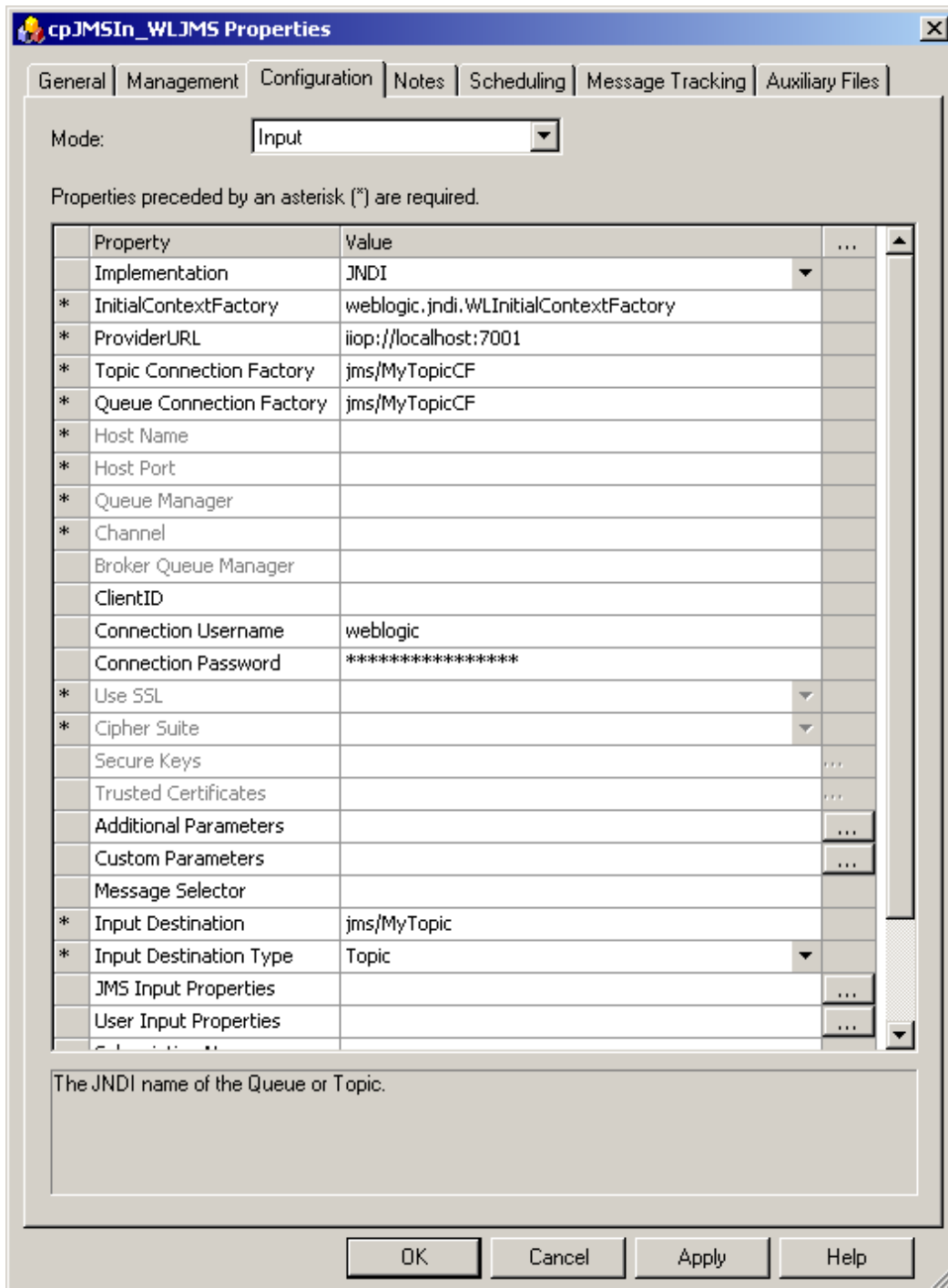
then its JMS Adapter must be correctly configured to support this communication, including making the wfullclient.jar available to the JMS Adapter. The following discuss the process.

Unlike what is shown in a number of articles which deal with configuring a JMS Client for integration with the WebLogic JMS, the protocol to use is not the “t3” protocol, but the “iiop” protocol. This is what works for Rhapsody. In summary, here are the key configuration options for Rhapsody 4.01 and their settings. Some settings, like the WebLogic JMS Host and JMS Destination name and type, will vary.

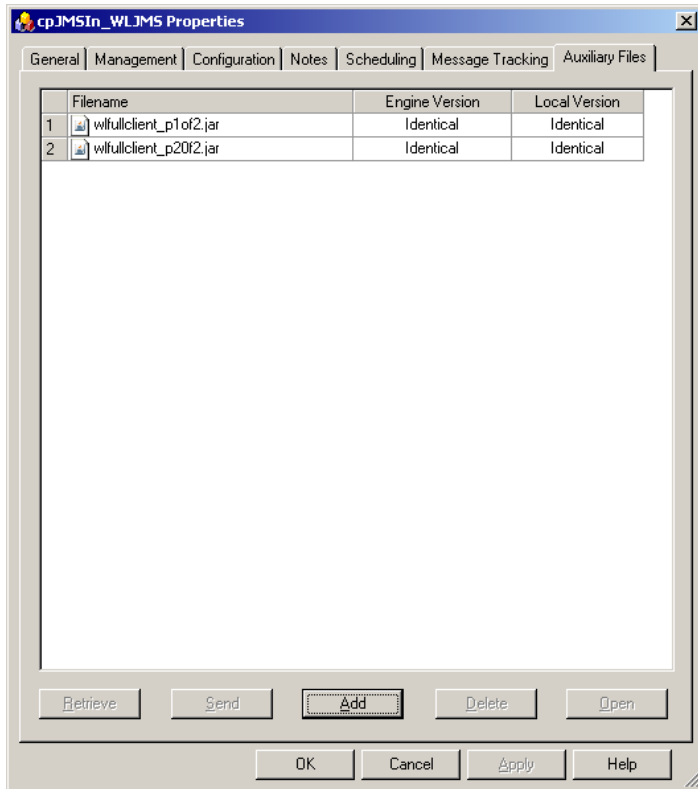
Property Name	Property Value
InitialContextFactory	weblogic.jndi.WLInitialContextFactory
ProviderURL	iiop://{WLJMSHost}:{WLJMSPort}
Topic Connection Factory	jms/{JMSTopicCopnnectionFactory}
Queue Connection Factory	jms/{JMSQueueConnectoinFactory}
Connection Username	{WLUsername}
Connection Password	{WLPassord}
Input Destination	jms/{MyTopicOrQueueName}
Input Destination Type	Topic (or Queue, depending on what you use)
Receiving Mode	Listening

In the table above values in {}, including {} themselves, must be replaced with the actual names from your configuration. For example, as shown in the picture below, my {WLJMSHost} would be replaced with localhost or some more appropriate host name or IP Address. My {WLJMSPort} would be replaced with 7001.

Rhapsody JMS Adapter configuration panel has an annoying idiosyncrasy where one must specify a Queue Connection Factory even if one uses a JMS topic. This does not have to be a Queue Connection Factory, as long as it is a connection factory. Without this the configuration cannot be saved.



The creation of a wfullclient.jar results in a JAR which is around 58Mb in size. The Communication Point wizard seems unable to deal with a JAR this size, when adding it through the "Auxiliary Files" tab. To get around this issue, copy the wfullclient.jar renaming the copy and the original so that you have two JARs – wfullclient\_p1of2.jar and wfullclient\_p2of2.jar. From the former delete the "weblogic" hierarchy, using 7-zip or some other archiver which can cope with JAR archives. From the latter delete everything \_except\_ the "weblogic" hierarchy. Add both archives to Auxiliary Files for the JMS Communication Point.



Now create a route with whatever additional communication points you might need and start the lot. Expect, if all is configured correctly, to receive messages from the topic hosted by the WebLogic JMS.

For the JMS objects, whose creation on the WebLogic Server side was discussed earlier, the settings would be:

Property Name	Property Value
InitialContextFactory	weblogic.jndi.WLInitialContextFactory
ProviderURL	iiop://localhost:7001
Topic Connection Factory	jms/MyTopicCF
Queue Connection Factory	jms/MyTopicCF
Connection Username	Weblogic
Connection Password	weclome1
Input Destination	jms/MyTopic
Input Destination Type	Topic
Receiving Mode	Listening

## Summary

In this article I walked through the process of setting up JMS Topic and its dependencies on WebLogic 10.3 platform and configuring Rhapsody 4.01 JMS Communication Point to receive messages from the JMS Topic hosted by the WebLogic 10.3 Server. Perhaps this will save you the time I spent figuring out how to do this.