Leading Open Source Middleware

OW2 Aspire RFID

Sub Project: RFID Suite

Administration Guide

Document	Administration Guide
Version	0.2
Last modification	24/08/07
Status	Final
Client	UJF/LIG/ADELE
Number of page	19
Authors	M2P GI : FORNACIARI François SURREL Guillaume VAUDAUX-RUTH Guillaume
Validated by	Team M2P GI DONSEZ Didier

TABLE OF REVISION

Version	Date	Modifications
0.1	07/08/07	Draft version
0.2	24/08/07	English corrections

TABLE OF CONTENTS

Table of revision 2

Table of contents 3

1. Introduction 5

2. Remote administration through JMX 5

2.1. JMX 5

2.1.1. Description 5

2.2. JConsole 5

2.2.1. Description 5

2.2.2. Connection 5

3. Make custom report 6

3.1. ECSpec documentation 6

3.2. JConsole screenshots 7

4. Application Level Events (ALE) 8

4.1. Build ECSpec with JConsole 10

5. Bundles list 11

6. Bundle configuration 17

6.1. Dispatcher 17

6.1.1. Description 17

6.1.2. Dispatcher.properties 17

6.1.3. JConsole screenshot 18

6.2. Mail pop reader 18

6.2.1. Description 18

6.2.2. mail.properties documentation 18

6.3. Export event mail 18

6.3.1. Description 18

6.3.2. mail.properties documentation 18

6.4. Mail server 19

6.4.1. Description 19

6.4.2. Conf.properties documentation 19

1. INTRODUCTION

This guide explains how to administrate the RFID Suite. Reading “**Installation guide**” before is strongly recommended.

2. REMOTE ADMINISTRATION THROUGH JMX

2.1. JMX

2.1.1. DESCRIPTION

JMX technology provides tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into those of the future.

2.2. JCONSOLE

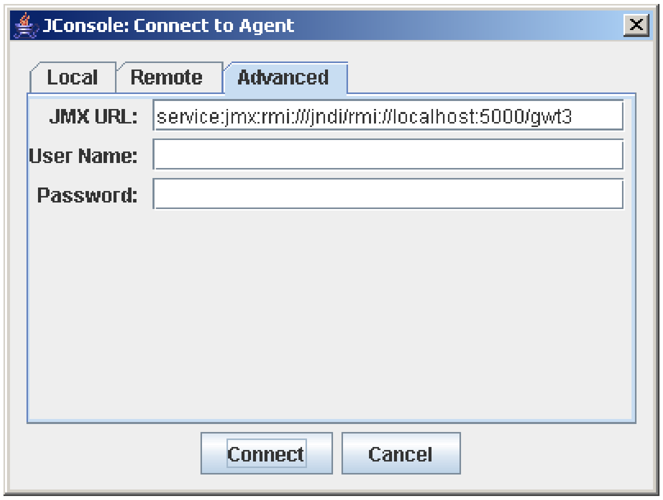
2.2.1. DESCRIPTION

JConsole is a JMX-compliant monitoring tool. It uses the extensive JMX instrumentation of the Java virtual machine to provide information on performance and resource consumption of applications running on the Java platform.

2.2.2. CONNECTION

Launch JConsole with:

```
> jconsole
```



JMX URL: service:jmx:rmi:///jndi/rmi://localhost:5000/gwt3

User Name:

Password:

Figure 1: JConsole connect screen

JMX URL Format: service:jmx:rmi:///jndi/rmi://**IP:Port/ProfileFelix**

IP : ip of the gateway which run Felix
Port : JMX port of the gateway

ProfileFelix : Felix profile name

No “User Name” and **no** “Password”

3. MAKE CUSTOM REPORT

To build a custom report you need to follow these steps:

1. Define what kind of information you want to send (ECSpec).
2. Define who receives this information (ALE).

3.1. ECSPEC DOCUMENTATION

Function	**addLogicalReaderName**(String *logicalName*)	
Description	Adds one name to the list of logical reader names which will be queried as a part of this ECSpec. If the name is already present, this method has no effect.	
Parameters	**logicalName**	the reader's logical name

Function	addECReportSpec(	String reportName, **boolean** includeEPC, **boolean** includeTag, **boolean** includeRawHex, **boolean** includeRawDecimal, **boolean** includeCount, String reportSetSpec, **boolean** reportIfEmpty, **boolean** reportOnlyOnChange, **boolean** includeGPS, **boolean** includeTemperature, **boolean** includeReaderName)
Description	Add an object which describes the individual report which will be generated by an event cycle.	
Parameters	**reportName**	the name of the report
	includeEPC	whether to include EPC in Pure Identity format in the report
	includeTag	whether to include EPC in tag format in the report
	includeRawHex	whether to include EPC in raw hexadecimal format in the report
	includeRawDecimal	whether to include EPC in raw decimal format in the report
	includeCount	whether to include the count of EPCs in the report
	reportSetSpec	define the EPCs to include. Can be **CURRENT**, **ADDITIONS** or **DELETIONS**. **CURRENT** is selected by default if the given string isn't recognized.
	reportIfEmpty	If set to true, the report should be generated even if it contains no EPCs.

		Otherwise, the no report will be generated in this case.
	reportOnlyOnChange	If reportOnlyOnChange is true, the ECReport is generated only if the set of EPCs, is different from the previous event cycle's set.
	includeGPS	whether to include GPS data in the report
	includeTemperature	whether to include temperature data in the report
	includeReaderName	whether to include the name of the logical reader

Function	**addECFilterSpec**(String reportName, String epcPattern, **boolean** isIncludingFilter)	
Description	Adds a filtering pattern to a report	
Parameters	**reportName**	the name of the report
	epcPattern	a URI object containing an EPC pattern.
	isIncludingFilter	if set to true, adds an including filter, otherwise adds an excluding filter.

Function	**addECGroupSpec**(String reportName, String epcPattern)	
Description	Adds a grouping pattern to a report	
Parameters	**reportName**	the name of the report
	epcPattern	a URI object containing an EPC pattern.

Function	**createECSpec**(String specName)	
Description	Links the created ECSpec to a name. Calls ALE.define(String specName).	
Parameters	specName	the name of the created ECSpec.

Function	**immediate**()
Description	Calls ALE.immediate method on the current ECSpec
Return	The immediate report

3.2. JCONSOLE SCREENSHOTS

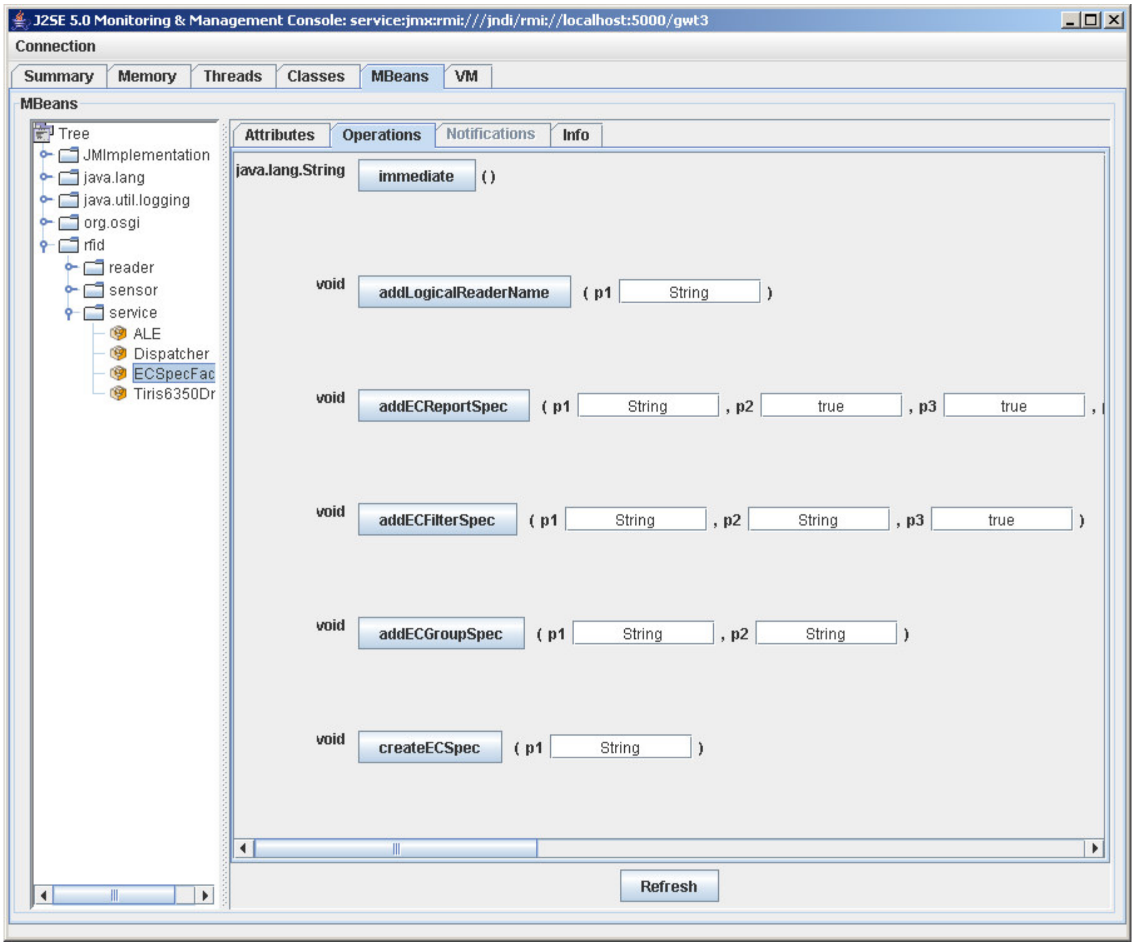

J2SE 5.0 Monitoring & Management Console: service:jmx:rmi:///jndi/rmi://localhost:5000/gwt3
Connection
Summary
Memory
Threads
Classes
MBeans
VM
MBeans
Tree
JMImplementation
java.lang
java.util.logging
org.osgi
rfid
reader
sensor
service
ALE
Dispatcher
ECSpecFac
Tiris6350Dr
Attributes
Operations
Notifications
Info
java.lang.String immediate ()
void addLogicalReaderName (p1 String)
void addECReportSpec (p1 String , p2 true , p3 true ,
void addECFilterSpec (p1 String , p2 String , p3 true)
void addECGroupSpec (p1 String , p2 String)
void createECSpec (p1 String)
Refresh

Figure 2: ECSpec JConsole screenshot

4. APPLICATION LEVEL EVENTS (ALE)

See http://www.epcglobalinc.org/standards/ale if you want more details about ALE.

Function	**void** undefine(String ecSpecName)	
Description	Undefine an event cycle specification. All existing subscriptions will be removed.	
Parameters	ecSpecName	The name of the specification to undefine.

Function	**void** subscribe(String ecSpecName, URI uri)	
Description	Subscribe to asynchronous report delivery from an event cycle specification.	
Parameters	ecSpecName	The name of the specification to subscribe to.
	uri	destination for asynchronously delivered reports.

Function	unsubscribe(String ecSpecName, URI uri)	
Description	Unsubscribe a specified destination from receiving asynchronous delivery of reports from a specified event cycle specification.	
Parameters	ecSpecName	The name of the specification to unsubscribe from.
	uri	The destination that will no longer receive reports.

Function	String poll(java.lang.String ecSpecName)	
Description	Activates a previously defined event cycle specification for one event cycle, synchronously returning a report.	
Parameters	ecSpecName	The name of the event cycle specification to activate.

Function	List getSubscribers(String specName)	
Description	Returns a list of URIs which are subscribed to asynchronous reports for the specified ECSpec name.	
Parameters	specName	The name of the event cycle specification

4.1. BUILD ECSPEC WITH JCONSOLE

Connect to your gateway with JConsole

1. See chapter 2.2.2

Create default ECSpec

1. addLogicalReaderName(fictive).
2. addECReportSpec, change only first parameter to:”myreport”
3. createECSpec(myecspec)

Define which computer will receive your report

1. Go to “Service>ALE”
2. Subscribe(myecspec, jms:fr.dyade.aaa.jndi2.client.NamingContextFactory/129.88.39.189/16400/129.88.39.189/16010/JTCF/rfidservertopic/root/root)

For more details about the URI destination format, see the info box below.

Now, your server receives **EMPTY** reports from your gateway by JMS.

Start fictive reader to fill in the report.

1. Go to “Readers>Fictive”
2. startReader()

Now, your server receives by JMS, reports **WITH INFORMATION** from your gateway.

URI destination must follow these formats:

SMTP	Format	**smtp**:mail server ip / mail server port / sender mail address / receiver mail address
	Example	**smtp**:129.88.39.189/25/homega@homega.fr/homega@homega.fr
WS	Format	**soap**: web service url
	Example	**soap**:http://129.88.39.189:8191/ReportServiceImpl?wsdl
JMS	Format	**jms:**NamingContextFactory**/**NamingContextIP**/**NamingContextPort**/**JMSServerIP **/**JMSServerPort**/**ConnectionFactoryBindingName**/**SendToTopicName**/**JMSLogin **/**JMSPassword
	Example	**jms**:fr.dyade.aaa.jndi2.client.NamingContextFactory/129.88.39.189/ 16400/129.88.39.189/16010/JTCF/rfidservertopic/root/root

5. BUNDLES LIST

If you want to launch a customized edge or premise, you need to have more information about the bundles. Here is a graph with the project dependencies and a descriptive table explaining the functionalities of the different jars:

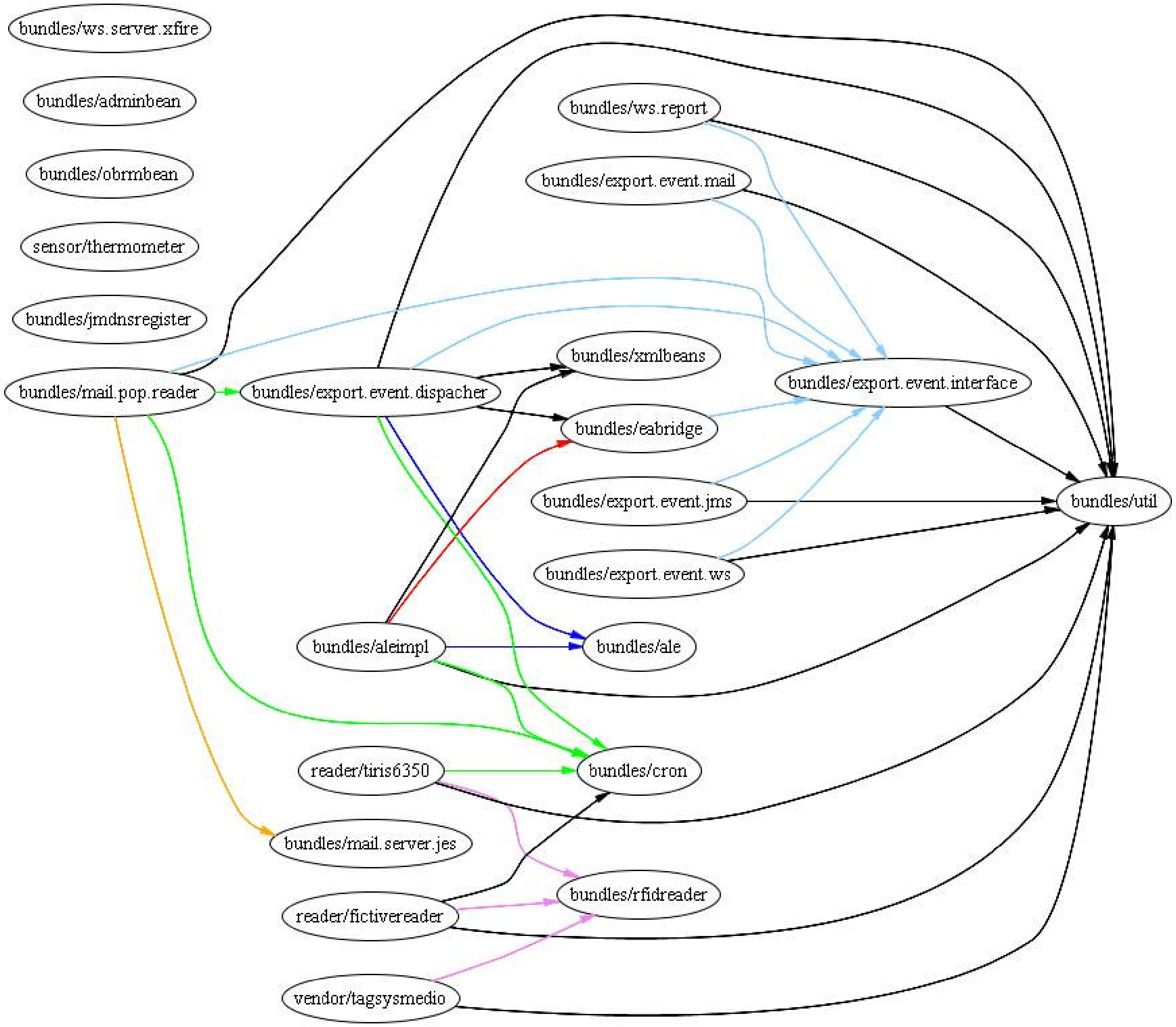


Figure 3: Package dependencies

Name	**aleimpl-0.1.0.jar**
Optional	False
Dependency	ale-0.1.0.jar
Description	This bundle implements the ALE specification version 1.0. ALE provides a standardized method to aggregate and filter data from readers. ALE defines also an XML schema (that can be extended) to allow standardized

	exchanges between many modules. If more than one business application needs to collect RFID information, the needed filters may be different for each application. In this case, we have two solutions: • create a filter instance for each application. Each filter sends its report to the corresponding application. In this case, the same information may be duplicated, and travel several times through the network. • The gateway filter is the sum of the needed filters, all needed information is collected, and stored to the database, and each application uses again its filter to interrogate the database. To be flexible on each solution, filters are generated, and configured to get either reader information, or other filter information in input. In the same way, the filter generates an ALE XML report, of an object structure which may be an input of another filter.

Name	**export.event.dispatcher-0.1.0.jar**
Optional	False
Dependency	eabridge -0.1.0.jar cron-0.1.0.jar org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar *Optional dependency :* export.event.jms-0.1.0.jar export.event.smtp-0.1.0.jar export.event.ws-0.1.0.jar
Description	Collected information needs to be sent to the premise or the server. This module dispatches messages through the protocol defined in them.

Name	**cron-0.1.0.jar**
Optional	False
Dependency	None
Description	It is a bundle that provides an OSGi^{TM} service to execute a task periodically. For example, it is used to send reports automatically.

Name	**export.event.jms-0.1.0.jar**
Optional	True
Dependency	export.event.interface-0.1.0.jar export.event.dispatcher-0.1.0.jar org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar
Description	This bundle provides the ability to send message by JMS protocol through the dispatcher.

Name	**export.event.ws-0.1.0.jar**
Optional	True
Dependency	export.event.interface-0.1.0.jar export.event.dispatcher-0.1.0.jar org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar
Description	This bundle provides the ability to send message by Web Service protocol through the dispatcher.

Name	**export.event.smtp-0.1.0.jar**
Optional	True

Dependency	export.event.interface-0.1.0.jar export.event.dispatcher-0.1.0.jar org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar
Description	This bundle provides the ability to send message by SMTP protocol through the dispatcher.

Name	**fictivereader-0.1.0.jar**
Optional	True
Dependency	eabridge -0.1.0.jar cron-0.1.0.jar org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar *Optional dependency :* Thermometer-0.1.0.jar
Description	This bundle simulates a RFID reader.

Name	**rfidreader-0.1.0.jar**
Optional	True
Dependency	None
Description	Export package of RFID readers services

Name	**adminmbean-0.1.0.jar**
Optional	False
Dependency	None
Description	Provides a service with a MBean interface to manage the OSGi gateway

Name	**mail.pop.reader-0.1.0.jar**
Optional	True
Dependency	cron-0.1.0.jar export.event.dispatcher-0.1.0.jar org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar
Description	A daemon which reads periodically the mail server specified in “mail.properties”.

Name	**mail.server-0.1.0.jar**
Optional	True
Dependency	None
Description	Java Email Server (JES)

Name	**tagsysmedio -0.1.0.jar**
Optional	True
Dependency	org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar com.tagsys.jar rxtx.jar *Optional dependency :* eabridge -0.1.0.jar
Description	Driver of TagSys RFID Reader.

Name	**tiris6350-0.1.0.jar**
Optional	True

Dependency	org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar rxtx.jar *Optional dependency :* org.apache.felix.eventadmin-0.9.0-SNAPSHOT.jar cron-0.1.0.jar
Description	Driver of Texas Instrument RFID Reader

Name	**util-0.1.0.jar**
Optional	False
Dependency	None
Description	A bundle which regroups useful functions

Name	**ws.report-0.1.0.jar**
Optional	True
Dependency	org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar *Optional dependency* export.event.dispatcher-0.1.0.jar ws.server.xfire-0.1.0.jar (without your service isn't exposed)
Description	Web service which receive report

Name	**ws.server.xfire-0.1.0.jar**
Optional	True
Dependency	util-0.1.0.jar
Description	Web service server, auto expose web service

Name	**JoramCommonsLibs-FELIX.jar-0.1.0.jar**
Optional	False
Dependency	None
Description	JORAM commons library.

Name	**joram-mom-bundle-FELIX.jar-0.1.0.jar**
Optional	True
Dependency	**JoramCommonsLibs-FELIX.jar-0.1.0.jar**
Description	JORAM server

Name	**org.apache.felix.mosgi.jmx.agent-0.9.0-SNAPSHOT.jar**
Optional	False
Dependency	org.apache.felix.log-0.9.0-SNAPSHOT.jar
Description	JMX server

Name	**org.apache.felix.mosgi.jmx.registry-0.9.0-SNAPSHOT.jar**
Optional	False
Dependency	org.apache.felix.log-0.9.0-SNAPSHOT.jar
Description	Starts the RMI registry

Name	**org.apache.felix.mosgi.jmx.rmiconnector-0.9.0-SNAPSHOT.jar**

Optional	False
Dependency	org.apache.felix.log-0.9.0-SNAPSHOT.jar
Description	Makes a connection between “JMX Registry” and “JMX Server”.

Name	**rxtx.jar**
Optional	True
Dependency	None
Description	Library to communicate with serial port

Name	**org.osgi.compendium-0.9.0-SNAPSHOT.jar**
Optional	False
Dependency	None
Description	Extends OSGi Core service

Name	**org.apache.felix.log-0.9.0-SNAPSHOT.jar**
Optional	True
Dependency	org.osgi.compendium-0.9.0-SNAPSHOT.jar
Description	Logger service

Name	**org.apache.felix.eventadmin-0.9.0-SNAPSHOT.jar**
Optional	False
Dependency	None
Description	OSGi event admin

Name	**jmdns.jar & jmdnsregister-0.1.0.jar**
Optional	False
Dependency	None
Description	Enables gateways auto discovery (for JASMINE console)

Name	**com.tagsys.jar**
Optional	True
Dependency	rxtx.jar
Description	Communication library for TagSys

Name	**org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar**
Optional	False
Dependency	None
Description	Tools providing easy management of bundle life cycle and dynamic service dependencies

Name	**eabridge-0.1.0.jar**
Optional	False
Dependency	org.apache.felix.ipojo-0.7.3-SNAPSHOT.jar *Optional dependency* export.event.dispatcher-0.1.0.jar org.apache.felix.eventadmin-0.9.0-SNAPSHOT.jar
Description	Transforms RFID Reader events into ALE events.

6. BUNDLE CONFIGURATION

6.1. DISPATCHER

6.1.1. DESCRIPTION

This OSGi bundle dispatches reports through three protocols in parallel or not:

- Web Service (WS)
- Simple Mail Transfer Protocol (SMTP)
- Java Message Service (JMS)

6.1.2. DISPATCHER.PROPERTIES

Documentation:

java.naming.factory.initial	JMS JNDI initial context name
java.naming.factory.port	JMS JNDI port
java.naming.factory.host	JMS JNDI ip
jms.topic.connection.factory.bind.name	Name of topic connection factory store in JMS JNDI
jms.server.host	Premise JMS Server ip
jms.server.port	Premise JMS Server port
jms.admin.login	Premise JMS Server admin login
jms.admin.pass	Premise JMS Server admin password
consumer.topic	Premise JMS consumer topic
destination.uri	**FORMAT:** **jms:**NamingContextFactory**/**NamingContextIP**/**NamingContextPort**/**JMSServerIP **/**JMSServerPort**/**ConnectionFactoryBindingName**/**SendToTopicName**/**JMSLogin **/**JMSPassword

Default config:

```
java.naming.factory.initial=fr.dyade.aaa.jndi2.client.NamingContextFactory
java.naming.factory.port=16401
java.naming.factory.host=129.88.39.189
jms.topic.connection.factory.bind.name=JTCF
jms.server.host=129.88.39.189
jms.server.port=16011
jms.admin.login=root
jms.admin.pass=root

#JMS topic consumer
consumer.topic=rfidpremisetopic

#URI used to send non treated reports and alerts
destination.uri=jms:fr.dyade.aaa.jndi2.client.NamingContextFactory/129.88.39.142/16400/
129.88.39.142/16010/JTCF/rfidservertopic/root/root
```

⚠ **DO NOT use localhost or 127.0.0.1 as the premise IP**

6.1.3. JCONSOLE SCREENSHOT

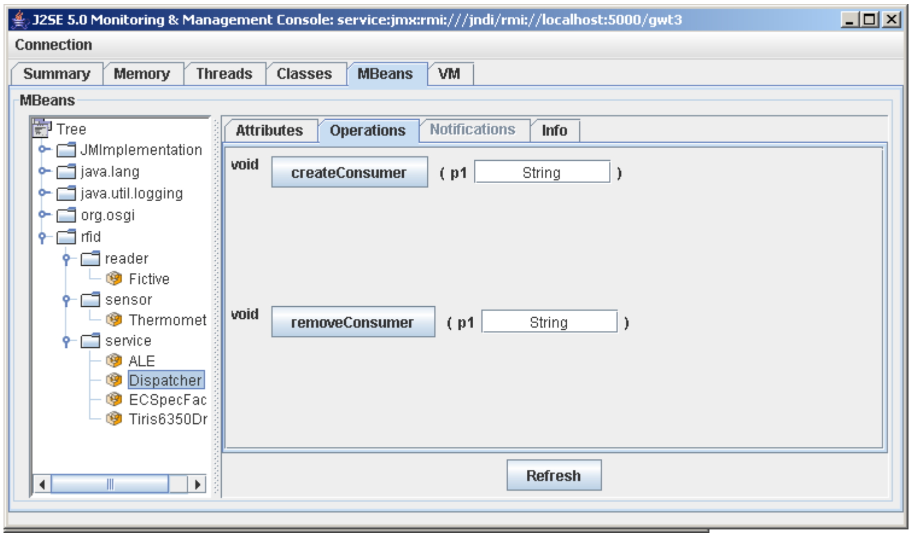


Figure 4: Dispatcher JConsole screenshot

6.2. MAIL POP READER

6.2.1. DESCRIPTION

The mail pop reader is a daemon which reads periodically a mail account with the POP3 protocol. You need to configure “**mail.properties**” with your own settings.

6.2.2. MAIL.PROPERTIES DOCUMENTATION

host	Mail server host ip
server	Mail server host port
account.name	Mail account name
account.login	Mail account password

6.3. EXPORT EVENT MAIL

6.3.1. DESCRIPTION

If you want to send messages through SMTP, you need to install “*export event mail*” and configure your “*mail.properties*” file.

6.3.2. MAIL.PROPERTIES DOCUMENTATION

mail.smtp.host	Mail server host ip
mail.smtp.port	Mail server host port

6.4. MAIL SERVER

6.4.1. DESCRIPTION

Each premise that provides SMTP transfer must embed a Java Email Server (*JES*), and of course each server need configuration.

6.4.2. CONF.PROPERTIES DOCUMENTATION

log.conf	Path to your JES log.conf
mail.conf	Path to your JES mail.conf
user.conf	Path to your JES user.conf
mail.directory	Path to your JES mail folder