PEtALS-BC-MAIL Component User's Guide

This document explain how to install and configure the petals-bc-mail JBI component.

PEtALS Team

Marie Sauvage <marie.sauvage@ebmwebsourcing.com>

Nicolas Salatge <nicolas.salatge@ebmwebsourcing.com>

- June 2007 -

Table of Contents

List of Figures

List of Tables

PEtALS-BC-MAIL

The Petals Mail binding component is a bidirectional binding component, it allows to :

- receive mails from external consumer and bind them to message exchanges intended to internal jbi components
- send mails to external services of Petals (internet addresses)

Chapter 1. Component Configuration

The following attributes can be set during the installation phase to configure the component, using the params element of the `jbi-install-component` ANT task:

no configuration for this component

Table 1.1. Component installation configuration attributes

Attribute	Description	Default	Required

Table 1.2. Advanced configuration of the component

Parameter	Description	Default	Required
pool-size	Number of threads listening to messages coming from the JBI container (JBIListeners). Int number >= 1	0	No
ignored-status	Status of messages exchanges that component must ignore. Accepted values : **DONE_AND_ERROR_IGNORED**, **DONE_IGNORED**, **ERROR_IGNORED** or **NOTHING_IGNORED**	DONE_AND_ERROR_IGNORED	No
jbi-listener-class-name	Fully qualified name of the class extending **AbstractJBIListener**		Yes
external-listener-class-name	Fully qualified name of the class extending **AbstractExternalListener**		No
properties-file	Name of the file containing values of keys used as reference by other parameters. To be able to configure a service-unit, you will use a key that has its value hosted by the component (ie. CDK documentation). The value of this parameter is : • whether an URL, • or a file relative to the directory defined by the environment variable PETALS_HOME.		No

Table 1.3. Interceptors configuration in the component

Parameter	Description	Default	Required
class	Name of the interceptor class. This class must extend the abstract class org.objectweb.petals.component.common.interceptor.Interceptor. This class have to be present in the classloader, in component or CF or in a shared library.		Yes
name	Name of the interceptor. This name will be used for additional configuration in the SU.	class name	No
active	Interceptor is active for all SU.	true	No

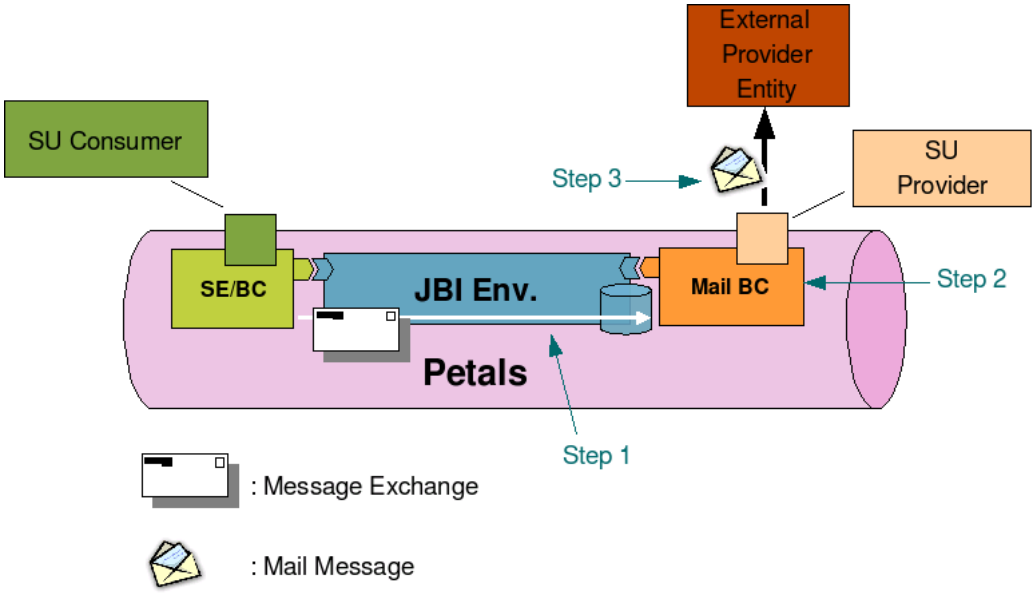
Chapter 2. Service Configuration

2.1. Send mails

PROVIDE SERVICE : Expose an external service in the JBI environment

Petals Mail binding component allows clients to send mails to external services of Petals (internet addresses). To send mails, MailBC can register new jbi endpoints and link them to specific email adresses (Internet Addresses). When MailBC receives a message exchange from Petals platform, it resolves the link to find the targeted email address, binds the message exchange to a mail message and send it to the recipient.

Figure 2.1. Sending mails



- Step 1 : A Consumer jbi component sends a Message Exchange to the Mail Binding Component.
- Step 2 : Mail Binding Component processes the Message Exchange : transforms it into a mail message and retrieve targeted External Provider Service (email address) linked to the endpoint set in the Message Exchange.
- Step 3 : Mail Binding Component sends this new mail to the targeted External Provider Service (Business Service, simple email account...).

2.1.1. Service Unit descriptor

Petals Mail binding component can be configured by deploying a new service unit to it. The jbi descriptor (`jbi.xml` file) of this service unit must contain a provides node describing the link between an internal jbi endpoint and an external email address. Here is an exemple of jbi descriptor activating a new "provided service" :

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi:jbi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:petals="http://petals.objectweb.org/extensions"
    xmlns:jbi="http://java.sun.com/xml/ns/jbi"
    version="1.0">
 <jbi:services binding-component="true">
  <jbi:provides interface-name="petals:MailInterface"
   service-name="petals:MailService" endpoint-name="MailEndpoint">
   <petals:wsdl></petals:wsdl>
   <petals:su-interceptors>
   </petals:su-interceptors>
   <petals:params>
    <petals:param name="scheme">smtp</petals:param>
```

```
    <petals:param name="hostname">Your email host name</petals:param>
    <petals:param name="username">Your user name</petals:param>
    <petals:param name="from">The email address from which the mail is sent</petals:param>
    <petals:param name="to">The destination email address</petals:param>
   </petals:params>
  </jbi:provides>
 </jbi:services>
</jbi:jbi>
```

Mail communication attributes :

Table 2.1. Service Unit attributes to provide services

Attribute	Description	Default Value	Required
provides	Name of the JBI service that will be activated to expose the Mail Destination into the JBI environment. interface (qname), service (qname) and endpoint (string) name are required.		Yes
scheme	the connection protocol (smtp)		Yes
username	the username used for authentication		No
password	the password used for authentication. Can be null or empty		No
hostname	the host address used for connection		Yes
port	the port used for connection	25 (default smtp port)	No
to	email address of the recipient **NEW FONCTIONNALITY** : You can specify the recipient address dynamicaly by setting a "destinationAddress" property in the incoming message exchange. **WARNING** : the recipient address must be specified at least once in the message exchange or in the SU descriptor		Yes
from	email address of the sender. This email address will be used as reply (from) address		No

Table 2.2. Advanced configuration of Service Unit (provides elements)

Parameter	Description	Default	Required
wsdl	path to a wsdl file describing services and operations offered by an endpoint activated by the SU. **This extension is only usable with provides fields.** The path can be a url "http" or "file" or relative to the r oot directory of the SU archive. Ex : "file:///user/ofabre/test.wsdl" or "/WSDL/test.wsdl" If no wsdl path is specified, a simplified description will automaticaly be written by the CF.		No

Table 2.3. Interceptors configuration in the Service Unit

Parameter	Description	Default	Required
name	Name of the interceptor to use. That's the name defined in the component.		Yes

2.1.2. Service Unit content

The Service Unit has to contain the following elements, packaged in an archive:

- The META-INF/jbi.xml descriptor file, has described above,
- An optional wsdl file describing the related service

```
service-unit.zip
+ META-INF
   - jbi.xml (as defined above)
   - service.wsdl (optional)
```

2.1.3. Usage

Once a *provides* node is configured, you can start to send email via the mail binding component. You just have to send message exchange to endpoints activated by service unit deployments (containing jbi.xml with provides node).

Caution

The subject of the sent mail is the name of the mail binding component. It can't be customized.

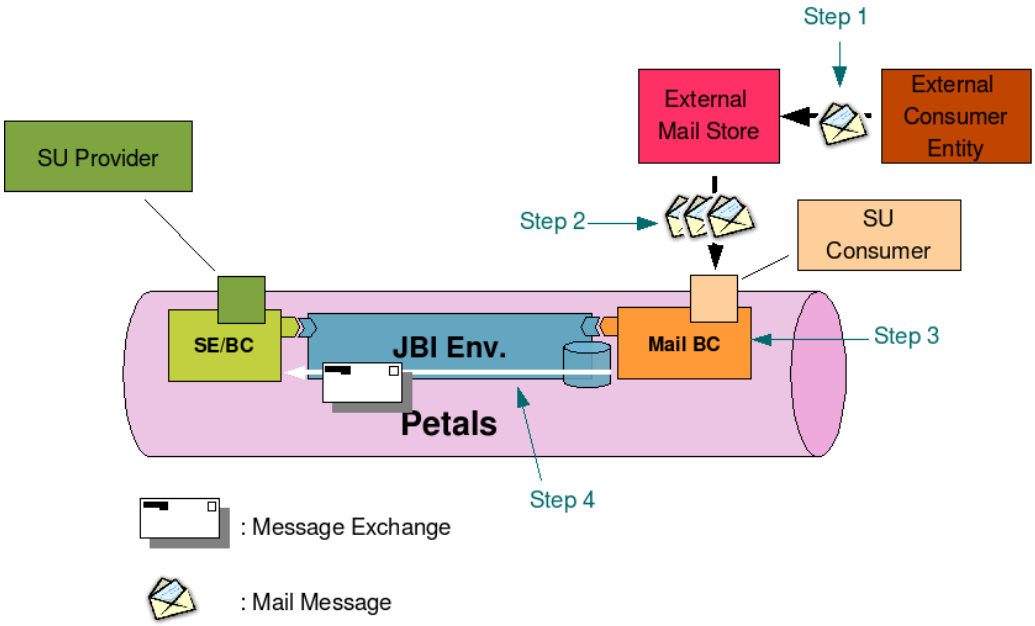
Caution

Only InOnly message exchange pattern is allowed.

2.2. Receive mails

CONSUME SERVICE : Expose an internal service outside the JBI environment

Figure 2.2. Receiving mails



Petals Mail binding component (MailBC) allows to receive mails from external consumer and to bind them to message exchanges intinded to internal jbi components. To receive new mails, MailBC can be linked to specific mail stores. It will check these stores periodicaly to retrieve new mails. If it finds a new mail in a store, it will process it (map this mail to a message exchange) and send it to the targeted jbi endpoint. Then the mail is removed from the store. So, all mails (read or unread) in a store are considered as new mail.

- Step 1 : An External Consumer Entity (Business Service or simple mail client) sends an email to the registered Mail Store (a classical email account).
- Step 2 : Mail Binding Component periodicaly checks for new mails and imports them.

- Step 3 and 4 : Mail Binding Component processes this new mails : transforms them into Message Exchanges, sends them to targeted jbi components (step 4) and finally delete them from the mail Store.

2.2.1. Service Unit descriptor

Petals Mail binding component can be configured by deploying a new service unit to it. The jbi descriptor (jbi.xml file) of this service unit must contains a consumes node describing the link between an external mail store and an internal jbi endpoint. Here is an exemple of jbi descriptor activating a new "consumed service" :

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi:jbi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:petals="http://petals.objectweb.org/extensions"
 xmlns:jbi="http://java.sun.com/xml/ns/jbi" version="1.0">
 <jbi:services binding-component="true">
  <jbi:consumes interface-name="petals:HelloWorldStaticInterface"
   service-name="petals:HelloWorldStaticService"
   endpoint-name="HelloWorldStaticEndpoint">
   <petals:mep></petals:mep>
   <petals:operation></petals:operation>
   <petals:timeout></petals:timeout>
   <petals:su-interceptors>
   </petals:su-interceptors>
   <petals:params>
    <petals:param name="scheme">Your email protocol (imap or pop3)</petals:param>
    <petals:param name="hostname">Your email hostname</petals:param>
    <petals:param name="username">Your user name</petals:param>
    <petals:param name="password">Your user password</petals:param>
   </petals:params>
  </jbi:consumes>
 </jbi:services>
</jbi:jbi>
```

Mail communication attributes :

Table 2.4. service-unit attributes to consume services

Attribute	Description	Default Value	Required
consumes	Name of the JBI service that will be called into the JBI environment. When a mail message is received. Only the interface (qname) name can be provided (the container will choose a ServiceEndpoint for this interface), or you can only set service (qname) and endpoint (string) names, without the interface name.		Yes
scheme	the connection protocol (imap or pop3)		Yes
username	the username used for authentication		No
password	the password used for authentication. Can be null or empty		No
hostname	the host address used for connection		Yes
port	the port used for connection	imap : 143 pop3 : 110	No
folder	the folder to check for new mails	INBOX	No
period	the checking period time	60 000 ms	No

Table 2.5. Advanced configuration of Service Unit (consumes elements)

Parameter	Description	Default	Required
mep	Message exchange pattern abbreviation. This parameter can be user in conjunction with a method of the Listeners : **createMessageExchange(Extensions extensions)**. This merthod returns a MessageExchange corresponding to the type of the specified pattern. Admitted values are : **InOnly**, **RobustInOnly**, **InOptionalOut** et **InOut**		Yes
operation	Operation to call on a service. This parameter can be used in conjuntcion with the sendXXX methods of the Listeners. If no operation is specified in the MessageExchange to send, this parameter will be used.		Yes
timeout	Timeout in milliseconds in a synchroneous send. this parameter can be used in conjunction with the **sendSync(MessageExchange exchange)** method of the Listeners. With this, a synchroneous send is done with this timeout value. 0 for no timeout int number >= 0 for a timeout	0	No
org.objectweb.petals.routing.strategy	This property defines the routing strategy. Two kind of strategy can be defines: highest or random. The others parameters represents respectively the local ponderation, the ponderation of the remote active endpoint and the ponderation of the remote inactive endpoint. The 'random' strategy chooses an endpoint in function of defined ponderations. The endpoints that have the strongest ponderation can be more easely choose in comparison with the others. The 'highest' strategy chooses the first endpoint in the list that have the strongest ponderation.		No
org.objectweb.petals.transport.compress	The payload of a MessageExchange is an XML file. It can be interesting to compress it before messages are exchanged between two PEtALS nodes. Values are `true` or `false`. True activated the compression of the content of the message.		No
org.objectweb.petals.mep.inbound.mustend	All MEP exchanges ended by a message containing a `DONE` or `ERROR` status. The consumer must accept those messages, otherwise they are accumulated in the NMR. Moreover, thoses messages cause useless trafic. Values are `true` or `false`. True make `DONE` or `ERROR` messages not sent.		No
org.objectweb.petals.transport.qos	This property set up the policy of the Quality of Service supported by Petals Transporter. Possible values are : reliable, fast. If not specified, the reliable policy is selected by default.		No

Table 2.6. Interceptors configuration in the Service Unit

Parameter	Description	Default	Required
name	Name of the interceptor to use. That's the name defined in the component.		Yes

2.2.2. Service Unit content

The Service Unit has to contain the following elements, packaged in an archive:

- The META-INF/jbi.xml descriptor file, has described above

```
service-unit.zip
+ META-INF
   - jbi.xml (as defined above)
```

2.2.3. Usage

Once an email store is configured, you can send email to it. You can specify an operation and a Message Exchange Pattern (MEP) to use with the targeted service. You just have to specify them in the email subject as follow : `operation=operation-name&mep=mep-name`

Table 2.7. Email subject attributes

Attribute	Description	Default Value	Required
operation-name	the targeted operation name		No
scheme	the message exchange pattern to use. The possible values are: in-only, robust-in-only, in-out, in-optional-out.  **Caution** Only InOnly pattern is supported for the moment. If no mep is specified, the default value is InOnly.		No

Chapter 3. Samples

Two usecases are described in this section:

- Sending Email Messages to the JBI Helloworld Service Engine.
- Sending JBI Messages to an external email box

This section describes how to install the different components and service assemblies to build these use cases.

In each case, the external JMS Server, where the client and provider queue are defined, must be started first.

3.1. Send an Email Message to the JBI Helloworld Service Engine

To send a Email Message to the JBI HelloWorld Service Engine, you must install several components in the order listed below:

- The HelloWorld Service Engine component (Download here).
- The Mail binding component (Download here).
- The sa-mail-consumes service assembly (Download here). This service assembly contains two service units:
  1. the su-mail-consumes service unit. This service unit consumes the endpoint defined by the next service unit.
  2. the su-helloworld-provides service unit.

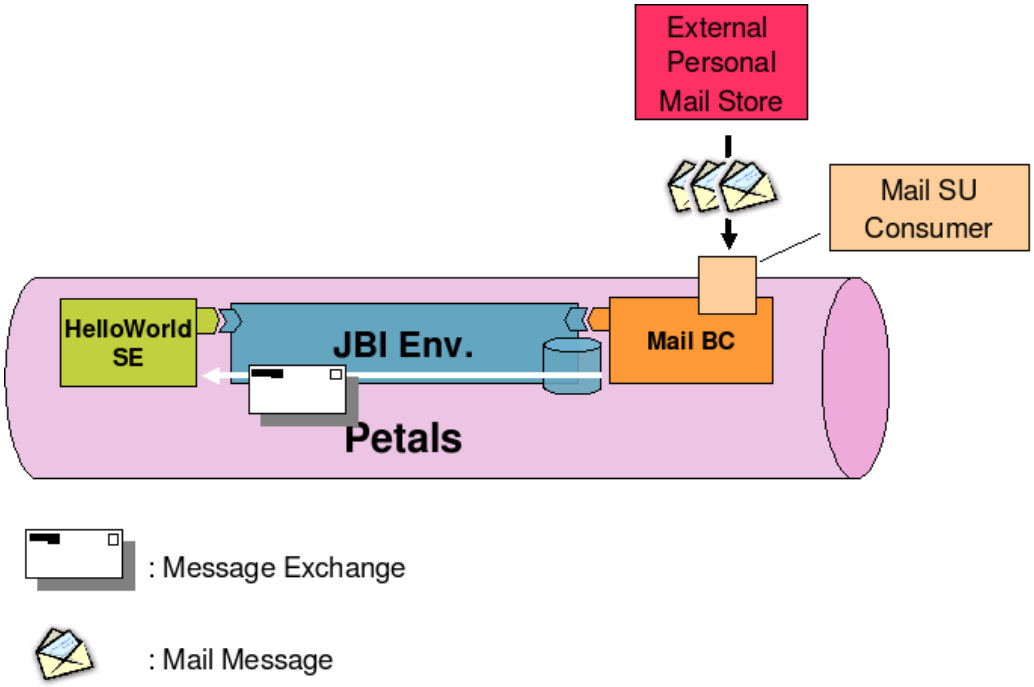
Caution

Configure the JBI descriptor of the su-mail-consumes service unit before installing it, to use your specific email server.

Caution

As the Mail component retreives and expunges all messages included in the extermal Email box to send them to the helloworld endpoint, it advice to use an dummy email account to realize this test.

Figure 3.1. The sa-mail-consume use case



3.2. Send a JBI Message to the external email box

To send a JBI Message to the external email box, you must install several components in the order listed below:

- The Sample Client Service Engine component (Download here).
- The Mail binding component (Download here).
- The sa-mail-provides service assembly (Download here). This service assembly contains one service unit: ths su-mail-provides service unit. This service unit provides an endpoint to send the email to the external email box.

Caution

Configure the JBI descriptor of this su-mail-provides service unit before installing it, to send email to your specific email address.

When the external email box is ready to receive messages, you can use the sample client to send xml message selecting the endpoint defined by the su-mail-provide service unit.

Figure 3.2. The sa-mail-provides usecase