PEtALS-BC-XMPP Component User's Guide

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

PEtALS Team
Maxime Carpentier <maxime.carpentier@ebmwebsourcing.com>

- June 2007 -

Table of Contents

PEtALS-BC-XMPP 5
1. Component Configuration 6
2. Service Configuration 7
2.1. Send XMPP messages 7
2.1.1. Service Unit descriptor 7
2.1.2. Service Unit content 9
2.1.3. Usage 9
2.1.4. Sending Files via XMPP 9
2.2. Receive Xmpp messages 9
2.2.1. Service Unit descriptor 10
2.2.2. Service Unit content 12
2.2.3. Usage 13
3. Samples 14
3.1. Sending a message to a jabber user 14
3.2. Sending a Xmpp Message to the JBI Helloworld Service Engine 15

List of Figures

2.1. Sending xmpp messages 7
2.2. Receiving xmpp messages 10
3.1. The sa-xmpp-provide usecase 14
3.2. The sa-xmpp-consume usecase 15

List of Tables

1.1. Component installation configuration attributes 6
1.2. Advanced configuration of the component 6
1.3. Interceptors configuration in the component 6
2.1. Service Unit attributes to provide services 8
2.2. Advanced configuration of Service Unit (provides elements) 8
2.3. Interceptors configuration in the Service Unit 9
2.4. Service Unit attributes to consume services 11
2.5. Advanced configuration of Service Unit (consumes elements) 12
2.6. Interceptors configuration in the Service Unit 12

PEtALS-BC-XMPP

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

- receive xmpp messages from external consumer and bind them to message exchanges intended to internal jbi components
- send text messages to an user connected to a jabber server
- send files to to an user connected to jabber server

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

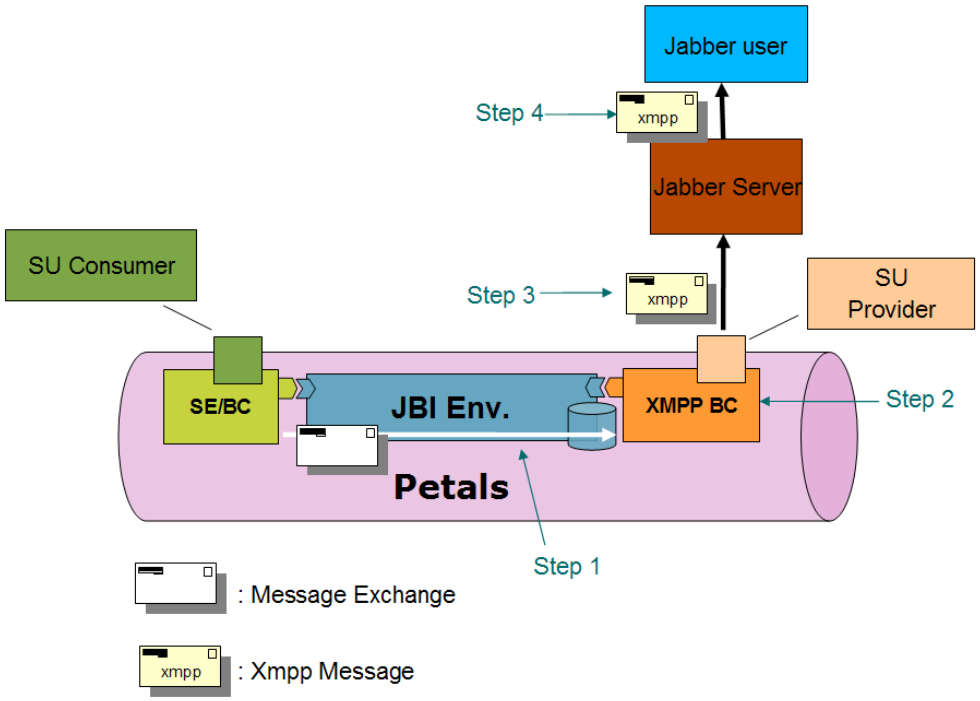
Chapter 2. Service Configuration

2.1. Send XMPP messages

PROVIDE SERVICE : Expose an external service in the JBI environment

Petals XMPP binding component allows clients to send text messages to a Jabber server. XmppBC register jbi endpoints with a server address, an user login/password . When XmppBC receives a message exchange from Petals platform, it connects to the registered jabber server with the registered user login/password (if not already connected), binds the message exchange to an xmpp message and send it to the server.

Figure 2.1. Sending xmpp messages



- Step 1 : A Consumer jbi component sends a Message Exchange to the Xmpp Binding Component.
- Step 2 : Xmpp Binding Component processes the Message Exchange : transforms it into a xmpp message and check if the connection to the xmpp server exists.If not, connect to this server.
- Step 3 : Xmpp Binding Component sends this new xmpp messages to the jabber server.
- Step 4 : The jabber server will then deliver the message to the user specified in the xmpp message

2.1.1. Service Unit descriptor

Petals Xmpp 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 the external jabber server. 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"
    xsi:schemaLocation="http://java.sun.com/xml/ns/jbi ../schema/jbi/standardPetals.xsd"
    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:xmppInterface"
   service-name="petals:xmppService" endpoint-name="xmppEndpoint">
   <petals:wsdl></petals:wsdl>
   <petals:su-interceptors>
   </petals:su-interceptors>
   <petals:params>
    <petals:param name="hostname">Address of the Jabber server</petals:param>
    <petals:param name="resource">Your user resource</petals:param>
    <petals:param name="username">Your user login</petals:param>
    <petals:param name="password">Your user password</petals:param>
    <petals:param name="default-destination">The default jabber address to send the message
 to</petals:param>
    <petals:param name="decoding">The char format encoding</petals:param>
   </petals:params>
  </jbi:provides>
 </jbi:services>
</jbi:jbi>
```

Xmpp 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 Xmpp server destination into the JBI environment. interface (qname), service (qname) and endpoint (string) name are required.		Yes
hostname	the address of the jabber server to connect to		Yes
username	the username used for authentication		Yes
resource	the resource for the user connection		No
decoding	the decoding char format	UTF-8	No
password	the password used for authentication.		Yes
default-destination	the default user to send message to, specified as a jabber address (<user>@<server>). Can be null or empty. Usually you specify the destination address in the message exchange, by specifying a "destinationAddress" property **WARNING** : the recipient address must be specified at least once in the message exchange or in the SU descriptor		Yes
transfert-timeout	the transfert timeout	60000	No
session-timeout	the session timeout	30000	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, as 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 xmpp messages via the xmpp binding component. You just have to send message exchange to endpoints activated by service unit deployments (containing jbi.xml with provides node).

Caution

Only InOnly message exchange pattern is allowed.

2.1.4. Sending Files via XMPP

Petals XMPP binding component allows clients to send files to a jabber user. All you have to do is to attach a file to your message exchange and it will be transfered as a file to the xmpp server

Caution

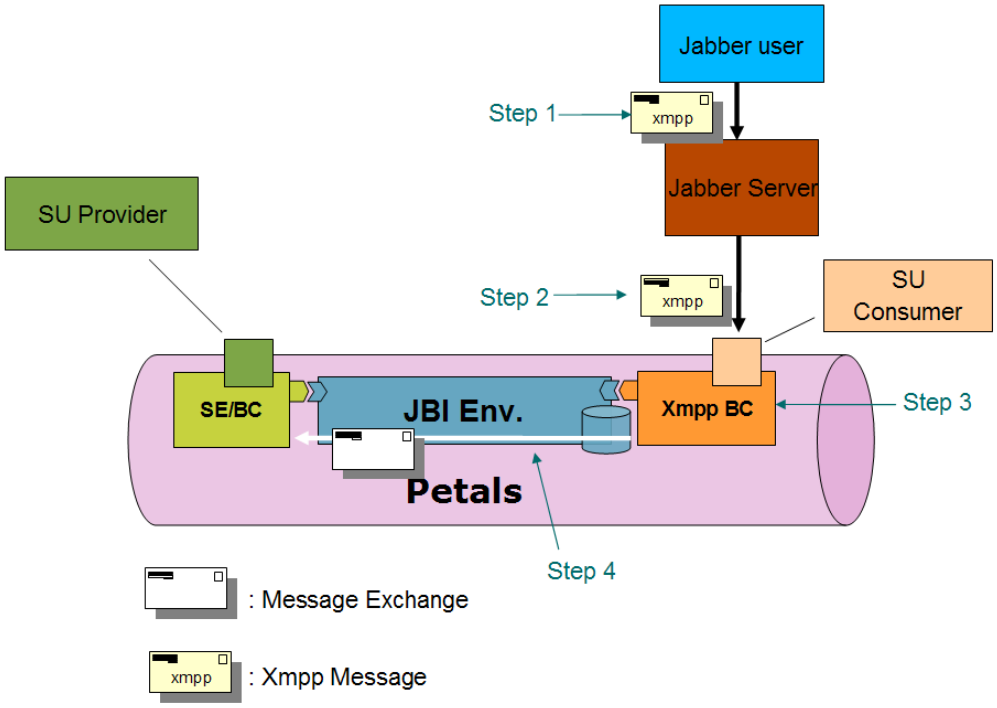
Multple attachments is not supported yet.

2.2. Receive Xmpp messages

CONSUME SERVICE : Expose an internal service outside the JBI environment

Petals Xmpp binding component (XmppBC) allows to receive xmpp messages from external consumer and to bind them to message exchanges intinded to internal jbi components. To receive new xmpp messages, XmppBC will connect to a specified jabber server, and listen to incoming messages.When receiving a new message,it will process it (map it to a message exchange) and send it to the targeted jbi endpoint. For now, only incoming text messages can be processed, all others messages (like presence or roster ones) are ignored.

Figure 2.2. Receiving xmpp messages



- Step 1 and 2: A Jabber client send an xmpp message to you via the jabber server.
- Step 3 : Xmpp Binding Component is connected to the jabber server and is listening to any text messages : the new message is received
- Step 4 : Xmpp Binding Component processes this new message : transforms them into Message Exchanges and sends them to targeted jbi components (step 4)

2.2.1. Service Unit descriptor

Petals Xmpp 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 jabber server 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"
 xsi:schemaLocation="http://java.sun.com/xml/ns/jbi ../schema/jbi/standardPetals.xsd"
 xmlns:petals="http://petals.objectweb.org/extensions"
 xmlns:jbi="http://java.sun.com/xml/ns/jbi" version="1.0">

 <jbi:services binding-component="false">
  <jbi:consumes interface-name="petals:soaportalInterface"
   service-name="petals:soaportalService"
   endpoint-name="soaportalEndpoint">

   <petals:mep></petals:mep>
   <petals:operation></petals:operation>
   <petals:timeout>0</petals:timeout>
   <petals:su-interceptors>
   </petals:su-interceptors>

   <petals:params>
    <petals:param name="hostname">The address of the jabber server</petals:param>
    <petals:param name="resource">Your user resource</petals:param>
    <petals:param name="username">Your user name</petals:param>
    <petals:param name="password">Your user password</petals:param>
```

```
    <petals:param name="pattern">The message exchange pattern</petals:param>
    <petals:param name="redirection">Jabber user address to force forward messages to</petals:param>
    <petals:param name="operation">The message exchange operation</petals:param>
    <petals:param name="encoding">The char format encoding</petals:param>
    <petals:param name="packets">The message packet</petals:param>
   </petals:params>

  </jbi:consumes>
 </jbi:services>
</jbi:jbi>
```

Xmpp 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 xmpp 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
username	the username used for authentication		No
password	the password used for authentication.		No
hostname	the host address used for connection		Yes
redirection	the user jabber address to force redirection		No
pattern	the message exchange pattern		No
operation	the message exchange operation		No
resource	the resource used	Home	No
packets	the message packets	Message	No
encoding	the char format encoding	UTF-8	No
decoding	the char format decoding	UTF-8	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.messaging.noack	All JBI exchanges are 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 is a property to 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 xmpp connection is configured, XmppBC will start listening for incoming text messages - just send messages via a simple jabber client to the reistered user and XmppBC will process it.

Caution

For now only text messages are processed, other messages (like rosters, presences, etc...) are ignored

Chapter 3. Samples

This section describes how to install the different components and service assemblies in order to test the XmppBC

For each usecase, you must:

- Have a jabber server running (for example, ejabberd) with at least 2 registered accounts.
- Have your jabber client (for example, Spark) connected to this server

3.1. Sending a message to a jabber user

To send a Xmpp Message to your jabber client, you must install :

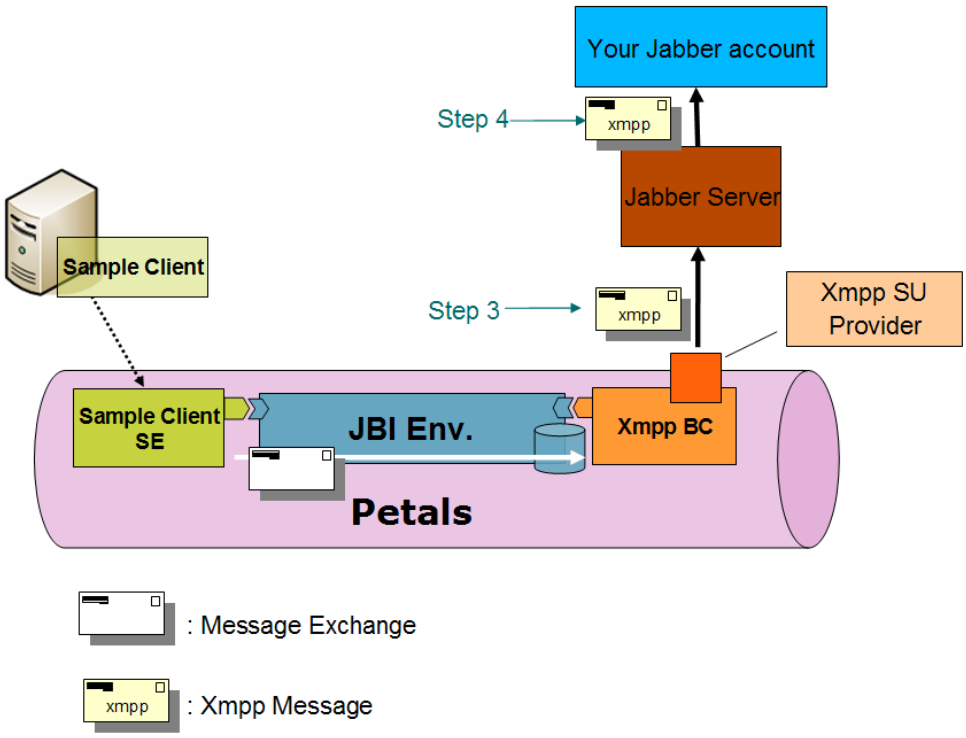
- The Xmpp binding component (Download here).
- The Sample Client Service Engine component (Download here).
- The sa-xmpp-provides service assembly (Download here). This service assembly contains the su-xmpp-provide service unit which provides an endpoint to send the xmpp messages to the external jabber server
  1. the su-xmppl-provide 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 this su-xmpp-provides service unit before installing it, to send messages to the right jabber server with the right account.

All these components can be found in Section 2.1, “Send XMPP messages”

Figure 3.1. The sa-xmpp-provide usecase



3.2. Sending a Xmpp Message to the JBI Helloworld Service Engine

To receive a message from a jabber server, you must install several components in the order listed below:

- The HelloWorld Service Engine component (Download here).
- The sa-helloworld-provides service assembly (Download here).
- The Xmpp binding component (Download here).
- The sa-xmpp-consume service assembly (Download here). This service assembly contains two service units:
  1. the su-xmpp-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-xmpp-consumes service unit before installing it, to use your specific jabber server and user/pwd.

All these components can be found inSection 2.2, “Receive Xmpp messages”

Figure 3.2. The sa-xmpp-consume usecase