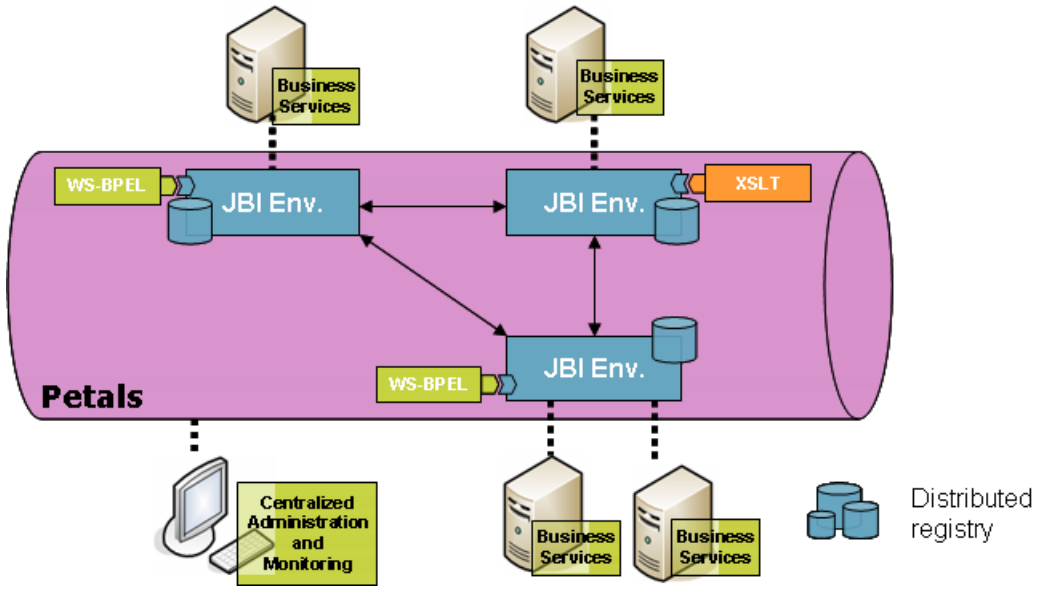
JOnES Demonstrator V2

In the first demonstrator, we have introduced some technical notions like BPEL, WSDL, process, (Web) services, and orchestration.
The aim of demonstrator V1 was to show the agility offered by BPEL orchestration.
We will now introduce a new software component into the architecture: an ESB Bus
by orchestrating webservice and business service exposed by an ESB, thanks to a BPEL engine.

The scenario :
I reserve a hotel and possibly a restaurant and I obtain a recapitulative e-mail with the sum to be paid.

What is PEtALS ?

PEtALS is the highly distributed Open Source ESB hosted by OW2. **PEtALS** delivers the OW2 Java TM Business Integration (JBI) platform.



To promote such an architecture, *PEtALS* implements the Java(tm) Business Integration specification (JSR 208). J.B.I. describes a "Service Component" approach.

Components are elements that offer services. Those Components are plugged on a **JBI container.** Services they expose are accessible through **Endpoints**.

Each consumer-component can request the JBI container to find a service. The JBI container gives to the consumer the corresponding Endpoint. Then, the consumer sends a message to the service-provider thanks to this Endpoint.

The JBI specification is strongly based on WSDL. Thus, each Component has to provide a WSDL description of the services it exposes. That's why PEtALS services are BPEL compliant.

PEtALS installation:

Information:

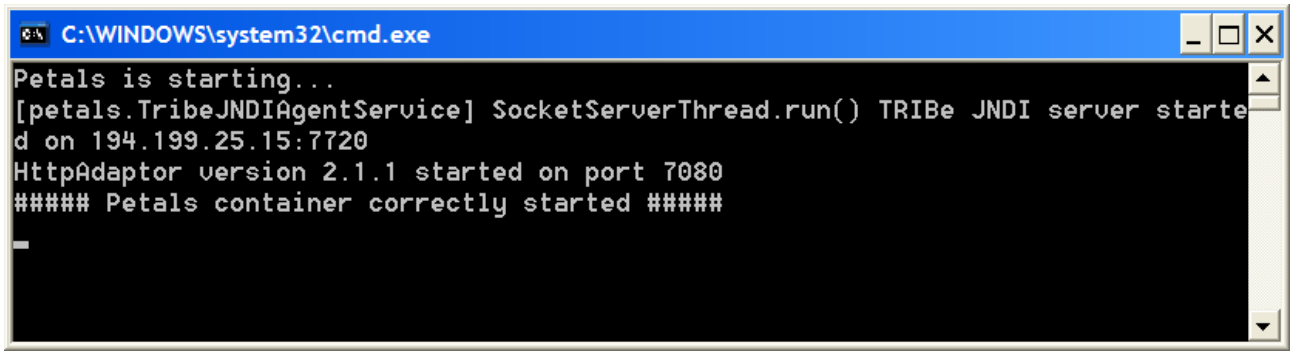
We are going to use functional version of the PEtALS which is 1.3
All the tests and information which will follow concern realizations made under Windows..

PEtALS 1.3 is downloadable at this address http://forge.objectweb.org/project/download.php?group_id=213&file_id=8401.
A directory named PEtALS-standalone-1.3 will appear after the downloaded archive will be unzipped, which we will henceforth name $PEtALS_HOME$.

Now, we have to create a PEtALS_HOME environment variable which will be logical name for the previously mentionned directory.

Now, you have to check if you can launch PEtALS by a double-click on $PEtALS_HOME $\bin\startup.bat (startup.sh under Linux)

You should obtain the window represent by the figure bellow :



Service exposition on PEtALS bus

In the archive named Composants_DemoJOnES_V2.zip which joins this documentation, you will find the following archives :

- PEtALS-bc-mail-1.2.zip
- PEtALS-bc-soap-1.3.zip
- PEtALS-se-mail-1.0.zip
- PEtALS-se-sampleclient-1.3.zip
- sa-consume-mailService.zip
- sa-provide-sendMailService.zip
- sa-provideMS-consumeSMS.zip

There are three types of components:

1. The binding components which make external link to application
2. The service engines which add functional service ti the bus
3. The service assembly which are the concrete service used by the demonstrator

We can notice that a binding component is reachable by two types of artifacts, the consume and the provide which allow different behaviour.

- Binding Components

The mail binding component allows to send and receive mails through PEtALS.
The soap binding component allows either to expose an external service as an endpoint on the bus or to expose an endpoint as an external webservice.

- Service Engines

The mail SE has been implemented in order to fill a missing functionnality of the PEtALS used version. Indeed, the communication inside the bus uses the MEP protocol which allows four possibilities : In-Only, In-Out, In-optional-Out, Robust-In-Only.
The soap-bc we use to call the mail service only allows In-Out while the bc-mail only supports In-Only calls. So we had to develop a component to transform the Message Exchange Pattern to make these two components compatible.
We can notice that since PEtALS 2.0, the problem doesn't exist anymore.

The SE SampleClient is provided on PEtALS distribution, and is a graphical client which allows to identify different service on the bus on to communicate with them.

- Services

There are the service we will use on the demonstrator which are not BC and SE

For more information on the specification, please read the official documentation about JBI (http://java.sun.com/integration/1.0/docs/sdk/) and PEtALS (http://PEtALS.objectweb.org/documentation.html) .

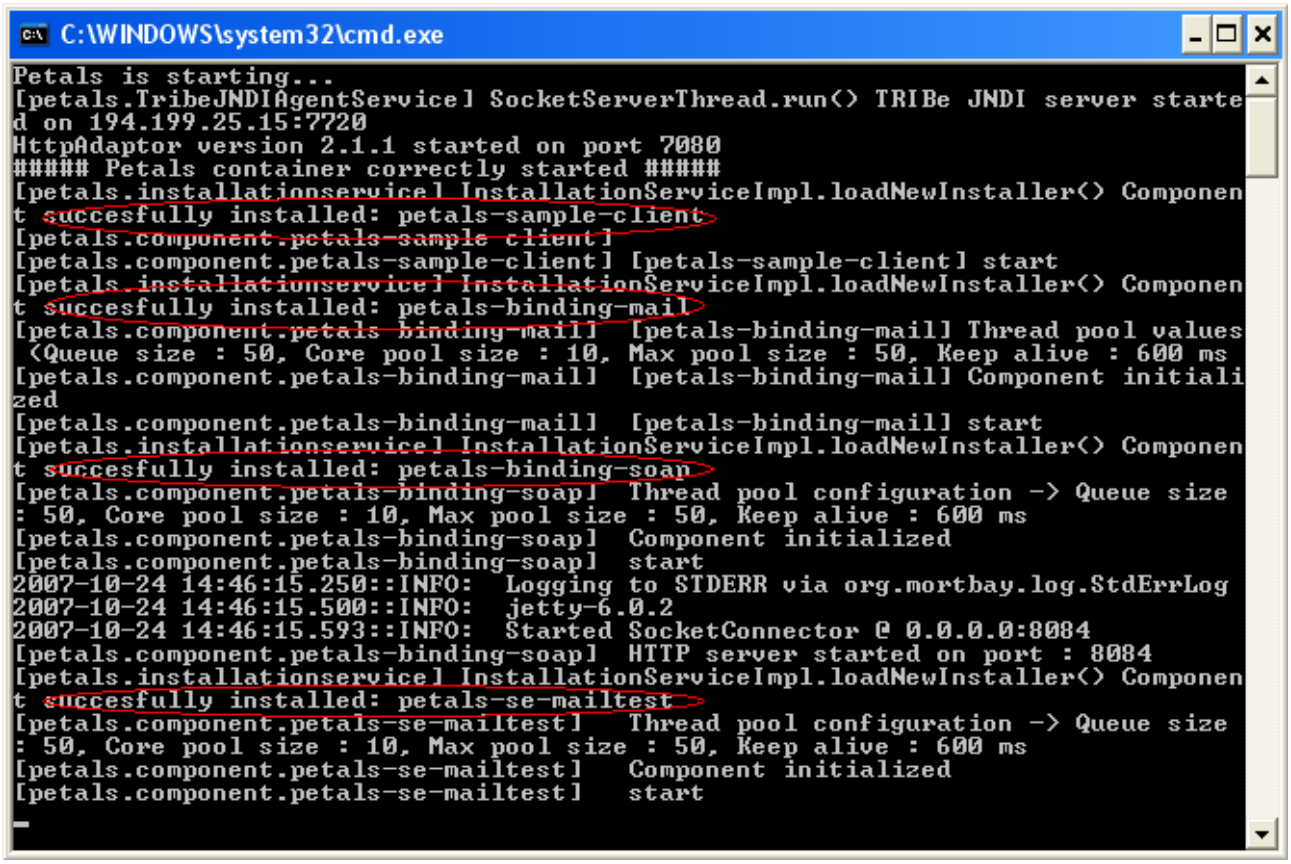
Archives deployment :

Let's detail the deployment of the need components and artifacts for our demonstrator.

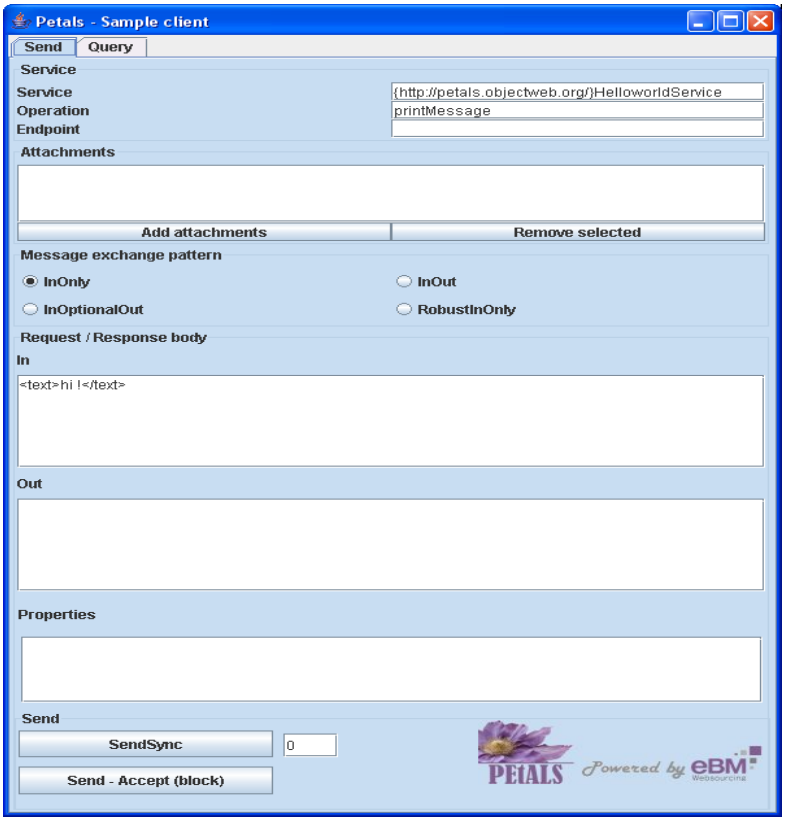
It's very easy to deploy a component on PEtALS : you just have to copy the archive to deploy and to paste it on $PEtALS_HOME$\install directory.

Let's copy the four following archives :PEtALS-bc-mail-1.2.zip, PEtALS-bc-soap-1.3.zip, PEtALS-se-mail-1.0.zip, et PEtALS-se-sampleclient-1.3.zip at the same time on the mentionned directory.

You may have the window graphically represented by the figure bellow where red surrounded parts show us the installation is successfull.



If the Se-client is successfully installed, this window should appear on your screen



Now, we have to deploy the following components one by one :

- sa-provide-sendMailService.zip

```
[petals.deploymentservice] DeploymentServiceImpl.deploy() Service Assembly succe
sfully deployed: petals-sa-mailbc
[petals.deploymentservice] DeploymentServiceImpl.start() petals-sa-mailbc succes
sfully started
```

- sa-provideMS-consumeSMS.zip

```
[petals.deploymentservice] DeploymentServiceImpl.deploy() Service Assembly succe
sfully deployed: petals-sa-helloworldtoext
[petals.deploymentservice] DeploymentServiceImpl.start() petals-sa-helloworldtoe
xt successfully started
```

- sa-consume-mailService.zip

```
[petals.deploymentservice] DeploymentServiceImpl.deploy() Service Assembly succe
sfully deployed: petals-sa-sendmail
[petals.component.petals-binding-soap]  Registering 'MailService' Axis service
[petals.deploymentservice] DeploymentServiceImpl.start() petals-sa-sendmail succ
essfully started
```

We can notice that for this last component deployment, we have this trace« Registering 'MailService' Axis service ». It means that the MailService EndPoint is reachable through an axis URI like http://HOST:port/axis2/services/MyService?wsdl

exp : http://192.168.1.10:8084/axis2/services/MailService?wsdl

WARNING : Services Assemblies can only be successfully deployed if binding-components and services-engines they depend on, are previously deployed. Furthermore, service deployment are intentionnally ordered because of the dependencies.
Thats's why the deployment order is important.
A bad deployment could make PEtALS crash.

Now, we have got three webservice, which are reachable through an URI:
Hotelboolking :
http://chlore.inrialpes.fr:8080/hotelbooking/celtix/hotelbooking
RestaurantBooking :
http://chlore.inrialpes.fr:8080/restaurantbooking/celtix/restaurantbooking
MailService
http://localhost:8084/axis2/services/MailService

Orchestration of Webservice with Orchestra

You just have to modify the URL on hotelbooking.wsdl and restaurantbooking.wsdl files located on $ORCHESTRA$\BPEL\samples\booking directory.

Hotelbooking.wsdl
Let's replace http://chlore.inrialpes.fr:8080/hotelbooking/celtix/hotelbooking by http://localhost:8084/axis2/services/HotelBookingService

```
.....
  <wsdl:service name="RestaurantBookingService">
    <wsdl:port name="restaurantbooking" binding="impl:restaurantbookingSoapBinding">
      <wsdlsoap:address location="http://chlore.inrialpes.fr:8080/hotelbooking/celtix/hotelbooking"/>
    </wsdl:port>
  </wsdl:service>
......
```

Restaurantbooking.wsdl
Let's replace http://chlore.inrialpes.fr:8080/restaurantbooking/celtix/restaurantbooking by http://localhost:8084/axis2/services/RestaurantBookingService

Then, you just have to delete and redeploy the booking service (provided with archive on wsdl_bpel directory) on orchestra and launch it in the same way than on demonstrator V1.
Reloading wsdl files by redeploying booking service, will allow to work with PEtALS exposed services.

Now that the orchestration is available, you have to use the webclient provided on an archive too.
The process to deploy it is the same than in Demonstrator V1.
Once your webclient application is launched, you may have this window :