PEtALS Quick Start Distribution

This document presents the Quick Start release of PEtALS. This release targets PEtALS beginners to ease their first step with PEtALS.

PEtALS Team
Roland NAUDIN <roland.naudin@ebmwebsourcing.com>
Christophe HAMERLING <christophe.hamerling@ebmwebsourcing.com>

- July 2008 -

Table of Contents

List of Figures

Preface

PEtALS delivers the OW2 Java Business Integration (JBI) bus (See JSR-208 specifications for further details on JBI).

Since its beginning, PEtALS has increased its coverage amongst different high value domains like clustering, robustness, availibility, performance... Moreover, PEtALS relies entirely on its OW2 partner technology, the Fractal component model, which brings to its architecture a strong modularity. Please visit the Fractal web site for further details at http://fractal.objectweb.org.

Since the version 2.1 of the PEtALS kernel, the PEtALS team have decided to exploit Fractal leverage by delivering various PEtALS distributions. Each distribution is packaged and customized to be addressed to a specific audience.

As its name suggests, this distribution is attended to PEtALS beginners, it proposes an all-in-one PEtALS container with an embedded Web console, simple use cases and this getting-started documentation.

The documentation leads you through the PEtALS universe and introduces the specialized documentations when needed.

Chapter 1. Presentation

1.1. What is PEtALS?

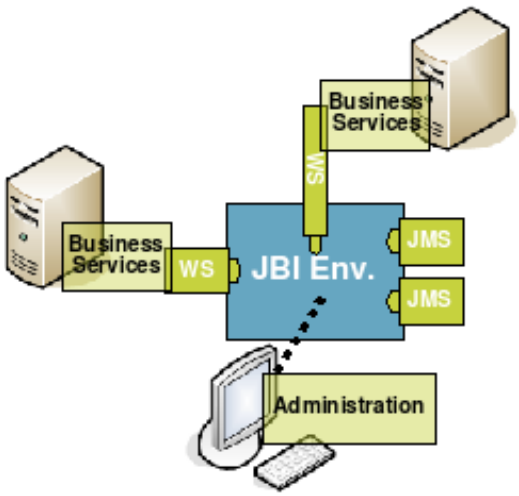
PEtALS helps you to integrate your Enterprise Business Units in order to provide a coherent and rational global solution. With PEtALS, you can compose your new applications by integrated already existing ones and new ones.

Thus, all your applications expose their business logic by exposing services. This concept is well known as Service Oriented Architecture (SOA).

PEtALS offers a solid backbone for your enterprise Information System and acts as a Bus, a place where all your data are exchanged. PEtALS connects services to each others. PEtALS is an Enteprise Service Bus (ESB).

Please visit our Web site for further details at http://petals.ow2.org.

Figure 1.1. PEtALS is an ESB fully JBI compliant



1.2. Acronyms

Along the document, we are using common JBI acronyms :

- **SL** : a Shared Libraries artifact.
- **BC** : a Binding Component artifact.
- **SE** : a Service Engine component artifact.
- **SA** : a Service Assembly artifact.
- **SU** : a Service Unit artifact.
- **MEP** : Message Exchange Pattern, JBI defines 4 possible MEP to invoke a service.

1.3. Quick Start features

The Quick Start distibution provides the following modules:

- **Standalone PEtALS container** : A standalone version of the PEtALS container, with an unique local transporter which transfers Message Exchange in memory (as reference). This version has been lighten of reliabilty, availibilty and clustering features.

- **Embedded Web Console** : The PEtALS Web Console embedded into the PEtALS container, which provides an enhanced GUI to monitor and administrate the PEtALS container. The Web Console is automatically started when PEtALS starts so you can access it at http://localhost:7878.
- **Sample Components** : A bunch of component samples helpfull to make any quick tests on the PEtALS Bus.
  - *petals-sample-client* : A graphical client to process quick visual tests based on JBI component API.
  - *petals-sample-clock* : A "ready at deployment" component which exposes a clock service automatically at its starting, without using the SA deployment mechanism.
  - *petals-sample-helloworld* : A simple component which exposes a helloworld service at the deployment of a basic SU contained in a SA.
- **Components** : A selection of components used along the Quick Start use cases.
  - *petals-bc-filetransfer* : A BC which permits to transfer files to and from the PEtALS Bus.
  - *petals-bc-soap* : A BC supporting Web Services. This component provides most of the features of WS-*, but we introduce only a simple usage in the Quick Start use cases.
  - *petals-se-xslt* : A SE which provides XSLT transformation features.
  - *petals-se-eip* : A SE bringing a library of integration patterns, uselfull to provide light orchestration.We use only a simple pattern in the Quick Start use cases.
- **Use Cases** : 4 use cases, fully documented, which guide you accross the PEtALS universe. Each use case introduce a new element to let you assimilate each important notions.
  - The *Clock* use case : This first use case makes you uses the Clock sample component and the Client sample component to invoke your first service in PEtALS.
  - The *Helloworld* use case : This use case lets you use the Web console to deploy the Helloworld sample component and its associated SA. The service invocations are still done with the Client sample component.
  - The *FileTransfer* use case : This use case introduces the usage of JBI dedicated ant tasks (http://ant.apache.org) and make you process file transfers with PEtALS.
  - The *Transformation* use case : This final use case introduces a more complex integration with a mini orchestration. It relies on the components SOAP, EIP and XSLT.

Once achieved the 4 use cases, you will be able to build your own integrations; you will be a new member PEtALS users community!

Chapter 2. PEtALS container

2.1. Pre-Requisites

- **Java** : To run PEtALS, you need at least a Java JRE 1.5 distribution.

  The Sun JVM can be downloaded at : http://java.sun.com/javase/downloads/index_jdk5.jsp.

- **Ant** : To run the provided use cases, you need to install an Ant distribution.

  Apache Ant can be downloaded at : http://ant.apache.org/bindownload.cgi.

2.2. Installation

PEtALS doesn't require any installer to be installed. Just unzip the provided archive where you want to install it.

Generally, PEtALS find by itself its directory location. On some systems, the installation path can not be automatically found, in such case you must set the *PETALS_HOME* environment variable :

- On Unix like system : `export PETALS_HOME=your_installation_path`.
- On Windows like system : `set PETALS_HOME=your_installation_path`.

2.3. Directory structure

- ant-samples : includes a sample file containing exemples to illustrate the use of JBI specialized Ant tasks.
- `bin` : includes scripts to launch PEtALS on Unix (*.sh) or Windows (*.bat) systems.
- components : contains the components required to run the Quick Start use cases.
- `conf` : includes PEtALS configuration files.
- `install` : auto-loader directory; put a JBI component (SE or BC) or a SA to have them automaticaly installed, deployed and started in the PEtALS container.
- `installed` : components and services assemblies are automaticaly copied into this directory after a succesful installation (or deployment).
- `lib` : includes all libraries required by PEtALS system.
- `logs` : includes all logs generated during PEtALS execution.
- `lost+found` : contains lost JBI elements (components or SAs no more referred in the PEtALS repository).
- `repository` : includes all the libraries, config files (etc) of the installed or deployed components, SLs and SAs.
- samples : contains the samples components required to run the Quick Start use cases.
- `schema` : includes some usefull XML schemas used by PEtALS.
- `uninstalled` : components, SLs and SAs are automatically moved into this directory after a succesful uninstallation (or undeployment).
- usecases : contains the usecases *Helloworld*, *FileTransfer* and *Transformation.*
- `webapps` : contains the Web console package to be deployed into Quick Start PEtALS.

- `work` : used by PEtALS system to put its intern resources during runtime.

2.4. Configuration

The Quick Start PEtALS is pre-configured to work properly.

If you want to customize the configuration, please use the PEtALS Standalone distribution and the PEtALS Platform distribution.

2.5. Starting PEtALS

1. Go to `$PETALS_HOME/bin`
2. Launch `startup.sh` or `startup.bat` :

```
# ./startup.sh
```

You can launch PEtALS in console mode, to be able to interact with it via the terminal. In this case, use the option -C in the terminal :

```
# ./startup.sh -C
```

Note

You can also use directlty the jar `server.jar` in the `$PETALS_HOME/bin`, with the command `start`, `stop`, `shutdown` as argument.

2.6. Stopping PEtALS

There is several ways to stop PEtALS :

1. If PEtALS is launched in a simple terminal, type **<ctrl>**+ **c** to stop PEtALS.
2. If PEtALS is launched in console mode, just type **q** or **x** and **<enter>** in the terminal :

```
# q
```

By typing **q**, PEtALS is stopped; components, SAs ant SLs aren't uninstalled/undeployed.

By typing **x**, PEtALS is shut down; components, SAs ant

SLs are uninstalled/undeployed.

3. If PEtALS is launched in background, launch the `stop.sh` script :

```
# ./stop.sh
```

4. If PEtALS is launched in background mode, launch the `shutdown.sh` script :

```
# ./shutdown.sh
```

Chapter 3. Use cases

3.1. Call the Time service

With this simple use case, you will be able with few manipulations to invoke your first service in the PEtALS Bus.

Here are the steps to follow :

- Start your PEtALS container (see section 2.5) in console mode
- Copy the Clock sample component located in the `samples` directory into the `$PETALS_HOME/install` directory . The PEtALS auto-installer automatically installs and starts the component
- In the PEtALS terminal, type `d`, the terminal will diplay the content of the PEtALS JNDI repository. You should see something the new endpoint exposed by the Clock component :

```
JNDI directory local
<== / ==>

        <=users=>

        <=new-endpoints=>
                -> {http://petals.ow2.org}ClockService@ClockEndpoint [Link to
 /endpoints/{http://petals.ow2.org}ClockService@ClockEndpoint]

        <=services=>

                <={http://petals.ow2.org}ClockService=>
                        -> {http://petals.ow2.org}ClockService@ClockEndpoint [Link to
 /endpoints/{http://petals.ow2.org}ClockService@ClockEndpoint]

        <=endpoints=>
                -> {http://petals.ow2.org}ClockService ->ClockEndpoint
 (INTERNAL):subdomain1/0/petals-sample-clock

        <=interfaces=>
```

- Copy the sample client component located in the `samples` directory into the `$PETALS_HOME/install` directory . A graphical interface is displayed at the component startup.
- With the help of the Client component documentation (http://petals.ow2.org/components.html), try to invoke the `printTime` operation to the `ClockEndpoint`. You should see a message on the PEtALS terminal like :

```
[petals.container.components.petals-sample-clock]-INFO 2008-07-18 14:48:24,923
 [petals-sample-clock] The ClockImpl service is called to display current time
[petals.container.components.petals-sample-clock]-INFO 2008-07-18 14:48:24,923
 [petals-sample-clock] Time : Fri Jul 18 14:48:24 CEST 2008
```

- Once your first service invocation done, try to change the MEP (Message Exchange Pattern) and the possible operations with the help of the service description (WSDL); see the error or fault generated when the operation or pattern is not supported by the Clock component.
- To clean your PEtALS server, type **x** on the PEtALS console. All the installed JBI artefacts are automatically uninstalled and the PEtALS container is shut down.

3.2. Say hello to the world

This use case introduces the usage of the PEtALS Web console. This console allows you to administrate and monitor you PEtALS container in a nice and intuitive way.

Follow the steps:

- Start your PEtALS container (see section 2.5)
- With your favorite Web browser (of course we preconize to use Firefox), connect to the Web Console at the URL http://localhost:7878. Use default values to configure the Data Collector and to log in.
- With the help of the Web Console documentation (http://petals.ow2.org/documentation.html), install and start the Helloworld sample component and the Client sample component, located in the `samples` directory.
- Once the components are installed, unzip the use case Helloworld, located in `usecases/petals-simplehelloworld.zip`. Deploy and start the contained SA named `sa-simplehelloworld.zip` with the Web console administration section.
- If you unzip the `sa-simplehelloworld.zip` SA, you will see that it contains another zip file, named `su-simplehelloworld-provide.zip`. This zip file is in fact a SU which decribes a provide service for the Helloworld component. Check in each JBI artifact the file `META-INF/jbi.xml`.This file is the JBI descriptor, the standard element which describes all the relevant information to build service configurations in JBI components.

  Note

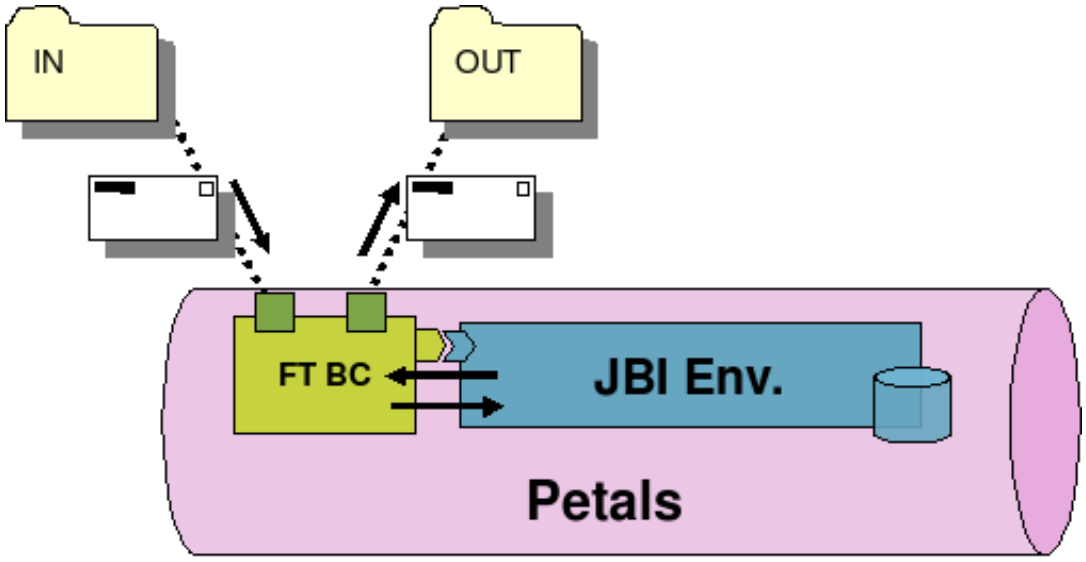
  If you check in the JBI components zip files, you will find the same JBI descriptor. In this case, the JBI descriptor defines the component configuration itself. If you want more information on these descriptors, please see the JBI specifications.

- With the Client sample GUI, get the service description (WSDL) of the Helloworld service and retrieve the available operations. Invoke the different operation (sayHello, printMessage) like for the previous use case and see what happened.
- To clean your PEtALS server, type **x** on the PEtALS console. All the installed JBI artefacts are automatically uninstalled and the PEtALS container is shut down. You cant just shut down PETALS by typing **q** for example. When you will restart PETALS, the installed JBI elements would be recovered to retrieve their previous life-cycle states.

3.3. Transfer a file

Now that you are more familliar with PEtALS and the sample components, we will introduce a real case of integration. This integration is quite simple, it consists to poll files from a local directory named `in` and transfer it to another directory named `out`. The file is sent through the PEtALS BUS as an attachment in a JBI Message Exchange.

Figure 3.1. File Transfer use case



In this use case, we introduce the usage of Apache Ant technology and specially the JBI dedicated ant tasks provided by PEtALS.

Note

You must have `Ant` installed on your system to process the use case (see section 2.1).

Please follow these steps:

- Unzip the File Transfer use case located at `usecases/petals-simplefiletransfer.zip`.
- In the use case exploded directory, you can find 3 directories and a file:
  - **deployables** : This directory contains all the JBI artifacts to install or deploy into PEtALS to run the use case.
  - **lib** : This directory contains the librairies required to process PEtALS Ant tasks.
  - **test** : This directory contains the resources required to process a test of the deployed use case.
  - **build.xml** : This XML file contains the definition of the Ant goals used to run the use case.
- For this use case, you must the PETALS_HOME system variable (see section 2.2).
- Start your PEtALS container.
- To process the use case, invoke the Ant goals in the following order from the use case root directory:
  - *prepare* : This goal prepares the environment to run the use case, it creates some directories in *$PETALS_HOME/usecases*.

```
# ant prepare
```

  - *deploy* : This goal installs, deploys and starts all the JBI artefacts required to run the use case.

```
# ant deploy
```

  - *run* : This goal runs the use case. It copies a file in the directory `$PETALS_HOME/usecases/simplefiletransfer/in` and wait that a file named `gettingstarted.xml` arrived in the directory `$PETALS_HOME/usecases/simplefiletransfer/out`, which represents the end of the integration processing.

```
# ant run
```

    In the PEtALS console, you should see log messages like :

```
[petals.container.components.petals-bc-filetransfer]-INFO 2008-07-18 16:23:25,760 process file :
 test-simplefiletransfer.xml
[petals.container.components.petals-bc-filetransfer]-INFO 2008-07-18 16:23:25,776 Process
 received exchange: petals:uid:B03895D355FFC23623175390829794572
[petals.container.components.petals-bc-filetransfer]-INFO 2008-07-18 16:23:25,780 Source
 transfer to file :
/home/chamerling/dev/petals/src/trunk/petals-distribution/petals-quickstart/target/petals-
quickstart-2.2-SNAPSHOT-dev.dir/petals-quickstart-2.2-SNAPSHOT/bin/./../usecases/
simplefiletransfer/out/gettingstarted-1216391005780.xml
[petals.container.components.petals-bc-filetransfer]-INFO 2008-07-18 16:23:25,876 Received
 acknowledgment 'Done' for transfered file
'/home/chamerling/dev/petals/src/trunk/petals-distribution/petals-quickstart/target/petals-
quickstart-2.2-SNAPSHOT-dev.dir/petals-quickstart-2.2-SNAPSHOT/repository/components/petals-bc-
filetransfer-07-18-2008-042201242/work/backup-test-simplefiletransfer1216391005761.xml'.
Delete it
```

    On the Ant client side, you should see a BUILD SUCCESSFUL message.

    In the `$PETALS_HOME/usecases/simplefiletransfer/out` folder you should see a new file. This file has been created from the JBI message received from the service consumer ie the file transfer component which has detected a new file in the input folder. The content of the output file is exactly the input one : You have transfered the file using PEtALS !

- *clean* : This goal cleans the PEtALS environment, it removes the deployed components, SA, and removes created directories.

```
# ant clean
```

You can have a look to the petals-bc-filetransfer component documentation (http://petals.ow2.org/components.html) for further details about how to configure it.

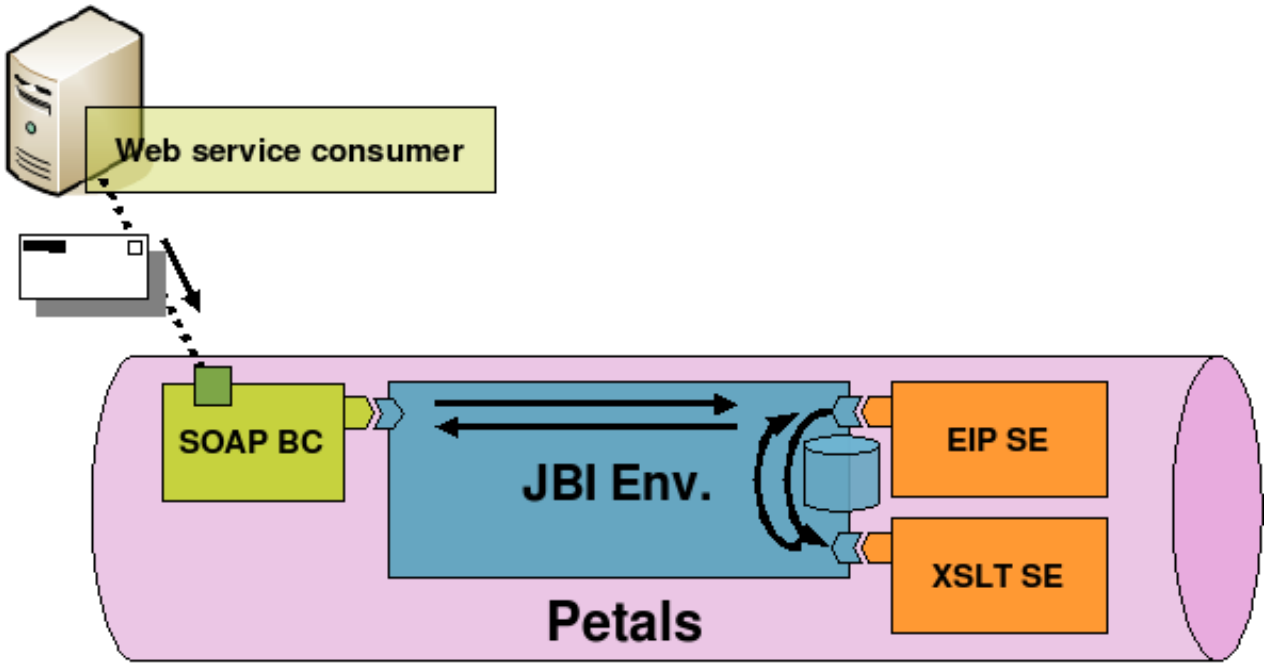
You can have a look to the samples of Ant task in the directory `ant-sample` of the Quick Start package, to have an illustration of all the possible PEtALS Ant tasks.

3.4. Make some data transformation

This last use case provides a more complex integration than an IN/OUT message flow. It involves a light orchestration of Message Exchange to build the complex message flows.

In this use case, we introduce the usage of the `EIP` component (Enterprise integration Patterns) to build a basic orchestration. The ingoing flow is exposed by a Web Service via the `SOAP` component, and the service itself is a transformation processed by the `XSLT` component.

Figure 3.2. Transformation use case



Here are the steps to follow for this usecase:

- Unzip the Transformation use case located at `usecases/petals-simpletransformation.zip`.
- Start your PEtALS container if it is not already started.
- Invoke the Ant goals in the following order from the use case root directory:
  - *deploy* : This goal installs, deploys and starts all the JBI artefacts required to run the use case.

```
# ant deploy
```

  - *run* : This goal runs a java program. This program is a Web Service client based on Axis2 which send a request to the transformation Web Service and controls if the returned document is correctly formatted.

```
# ant run
```

On the Ant client side, you should see this type of message :

```
run:
     [echo] Launching usecase simpletransformation...
     [java] log4j:WARN No appenders could be found for logger (org.apache.axis2.util.Loader).
     [java] log4j:WARN Please initialize the log4j system properly.
     [java]
     [java] Invoking the Web Service at
 'http://localhost:8084/petals/services/ExternalSimpleTransformation' with payload:
     [java] Payload:
     [java] <simpletransformation:request
 xmlns:simpletransformation="http://petals.ow2.org/simpletransformation">
              <simpletransformation:name>NAUDIN</simpletransformation:name>
              <simpletransformation:street>Paradise Street</simpletransformation:street>
              <simpletransformation:city>Toulouse</simpletransformation:city>
              <simpletransformation:documentid>5040</simpletransformation:documentid>
            </simpletransformation:request>
     [java]
     [java] Response payload:
     [java] <simpletransformation:document
 xmlns:simpletransformation="http://petals.ow2.org/simpletransformation"
 simpletransformation:id="5040">
     [java]   <simpletransformation:name>NAUDIN</simpletransformation:name>
     [java]   <simpletransformation:area>Toulouse - Paradise Street</simpletransformation:area>
     [java] </simpletransformation:document>
     [java] Usecase simpletransformation SUCCESSFUL
```

On the PEtALS console, you will see some log messages which shows that the SOAP message has been received by the SOAP BC, forwarded to the EIP one, to the XSLT...

- *clean* : This goal cleans the PEtALS environment, it removes the deployed components and SA.

```
# ant clean
```

Do not hesitate to pick a glance to the SA package in the `deployables` directory of the use case. You can see how the SUs are configured compared to the related component documentations (http://petals.ow2.org/documentation.html).

Note

If you want to test this use case with a graphical Web Service client, please have a look to the SOAP use cases documentation to get information about how using `SOAPUI` product.

Chapter 4. Next steps...

Now that you are quite familiar with PEtALS, you can start creating your own integration or service oriented application based on the large repository of PEtALS components. Check the available components at http://petals.ow2.org/components.html.

More adavanced use cases mainly based on SOAP capabilities are availbale on the project Web site.

If you want to control or optimize the PEtALS configuration, please refer to the Standalone distribution documentation.

If you are interesting about running PEtALS in a distributed environment, please refer to the Platform distribution documentation.

You can be interested to build your own components and thus join the PEtALS community. The PEtALS team provides a framework which handles most of the parts of JBI mechanisms and lets you focus on your business logic: the Component Development Kit (CDK).

PEtALS team work is based on prooved quality tools. The team has also written a guide to lead any interested developer who wish to join the PEtALS developer team (http://petals.ow2.org/documentation.html).