



*SLAs Empowering a Dependable
Service Economy*



ORC Deployment Guide

Whitepaper

Version 1.0 July 21, 2011

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216556.

Contributors	
Partner	Contributors
FZI	Christoph Rathfelder, Carolin Gärtner

Notices
<p>The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2010 by the SLA@SOI consortium.</p>
<p>* Other names and brands may be claimed as the property of others.</p>



This work is licensed under a [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/).

Executive Summary

The Open Reference Case (ORC) is used as an open source demonstrator highlighting features provided by the SLA@SOI framework. The ORC demonstrates the use of the SLA management framework in the context of a service-oriented retail solution supporting the sales process in supermarkets. In this document we describe the installation of the ORC on Ubuntu based. The ORC can be setup in different deployment options.

Table of Contents

1	Introduction	5
2	Prerequisites	7
3	Manual Installation from Source.....	7
3.1	Installation of Management Interface and BPEL Engine	9
4	Configuration.....	9
4.1	Manual Configuration.....	9
4.1.1	Start the ORC manually	10
4.1.2	Manual Deployment and Undeployment of Services	11
4.2	Configuration Services.....	11
4.2.1	Configuration Service	11
4.2.2	Bank Configuration Service	12
5	Testing the ORC.....	12
5.1	PaymentService.....	13
5.2	InventoryService	14
5.3	CompositeService	14

1 Introduction

The Open Reference Case (ORC) is used as an open source demonstrator highlighting features provided by the SLA@SOI framework. The ORC demonstrates the use of the SLA management framework in the context of a service-oriented retail solution supporting the sales process in supermarkets. In more detail, the ORC includes IT support for retail chains in general, covering enterprise headquarters (central management issues), stores (local management) and cash desks. Several shop providers, each with a certain number of stores, are connected to a single service provider, supporting sales of goods with an IT system. The ORC software can run on a virtualized infrastructure using various deployment options, to cover small as well as large shops with different requirements regarding the performance of the system.

This documentation serves to describe the deployment details including necessary system configurations of the Open Reference Case (ORC) demo on Ubuntu, which is a free Enterprise-class Linux Distribution. The ORC software solution can be deployed on one single machine as well as distributed across up to three machines. This allows selecting the most appropriate option depending on the additional requirements to the functionality, such as the completion time or cost depending on the requirements and the expected amount of customers in the shop. It is also possible to use the ORC as a multi-tenant system that handles several different shops on one single ORC instance. Figure 1 illustrates some of these deployment options. Although the services are mostly independent, there are some constraints limiting the deployment options. The three services `InventoryService`, `StoreInformationService`, and `OderService` must be deployed on the same machine, as they are services wrappers for the software components running in the back-end, which have some shared functionality and configuration. The `CardValidationService` and the `PaymentDebitService` are also bundled as a single deployment unit, since separating the mechanisms for card-validation and debiting makes no sense as these are always provided by the same organization. In the SLA@SOI framework, we use virtual machines, which allows us to deploy new service instances by cloning and starting existing VM images. This dramatically reduces the effort compared to manual installation on physical hosts. However, it is still possible to manually install the ORC without virtualization if required. The different deployment options that can be realized are described in the following and visualized in Figure 1.

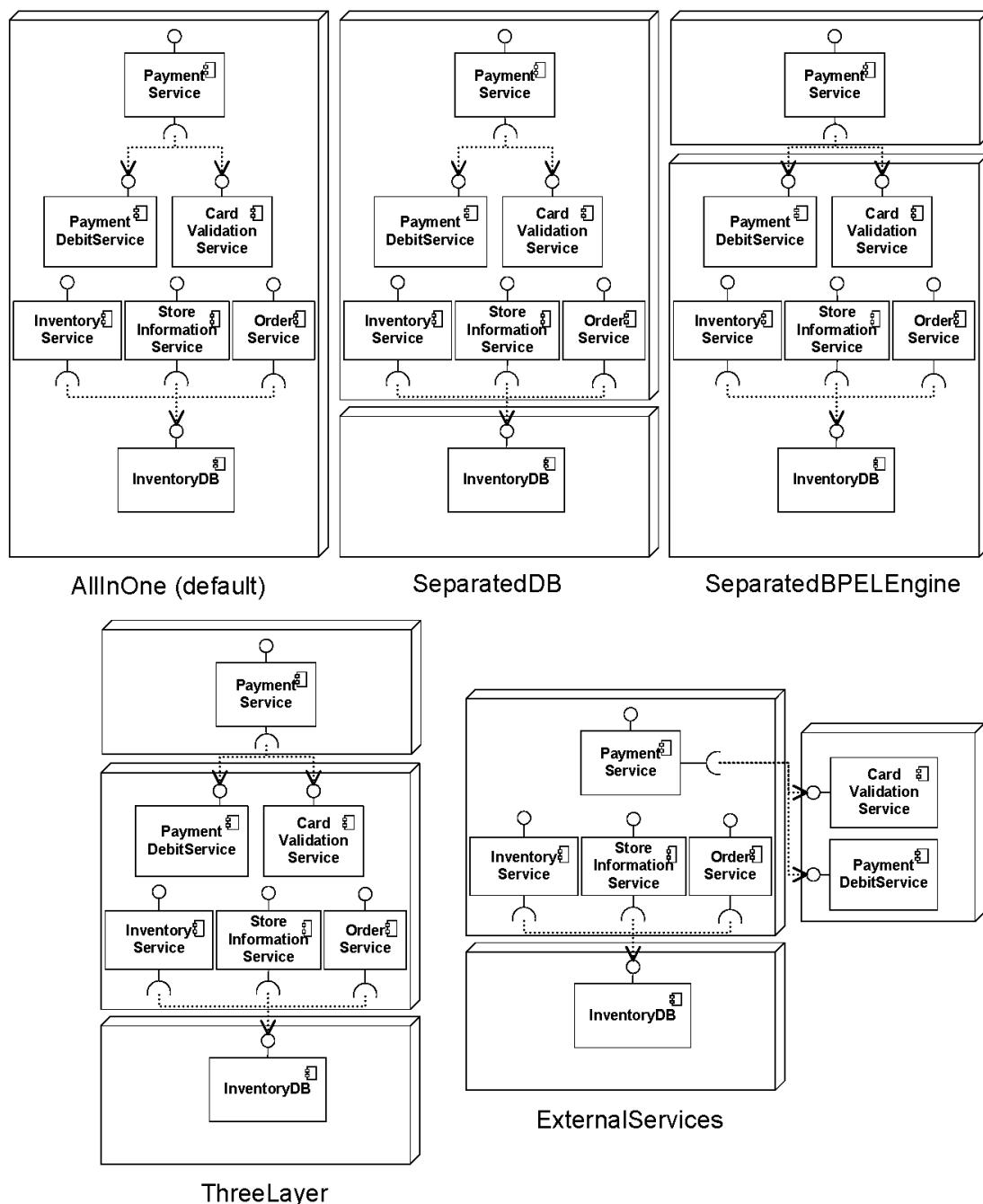


Figure 1: ORC Deployment Options

AllInOne: This deployment option consists of only one single virtual machine image. This image contains the database as well as the basic services and the composite services which include the application logic of the ORC.

SeparatedDB: The second bundle is a virtual appliance consisting of two virtual machine images. One provides an application server with all deployed services and the other hosts only the database.

SeparatedBPELEngine: The DB consumes only a small part of the available CPU resources. Most CPU power is consumed in the web service container and the BPEL engine. Therefore this deployment option separates the BPEL engine on an

individual virtual machine. Basic Services and the database are deployed on another virtual machine.

ThreeLayer: This deployment option is a combination of the two previous ones. Database as well as BPEL engines are separated from the basic services. So this deployment option consists of three VMs.

ExternalServices: ORC can also be used to demonstrate the involvement of an external service provider. It is possible to transfer the two services `CardValidationService` and `PaymentDebitService` to an external virtual machine. By this, they can be used as if they were provided by an external service provider including the negotiation of SLAs with this external service provider.

2 Prerequisites

- Ubuntu 10.x
- Java Development Kit (JDK) 1.5 or higher
- Apache Ant 1.6 or higher
- No white spaces in the installation path of the system

Note that so far the ActiveBPEL Engine Community Edition only supports JDK 1.5. Therefore, the Tomcat 5.5 is chosen the servlet container. For its commercial version, ActiveVOS Engine already supports JDK 1.6 and Tomcat 6. The ActiveVOS Designer has an internal ActiveVOS Engine 6 for debug use as an optional. However, according to the project requirement, the open-source ActiveBPEL Engine Community Edition is preferable.

3 Manual Installation from Source

1. Check out the ORC source code from SourceForge.
2. Copy the following JAR files to the `lib` folder of the ORC project:
 - a. From Axis 1.4 (http://www.apache.org/dyn/closer.cgi/ws/axis/1_4/):
 - i. `axis.jar`
 - ii. `commons-discovery-0.2.jar`
 - iii. `commons-logging-1.0.4.jar`
 - iv. `jaxrpc.jar`
 - v. `log4j-1.2.8.jar`
 - vi. `saaj.jar`
 - vii. `wsdl4j-1.5.1.jar`
 - b. From Derby (http://db.apache.org/derby/derby_downloads.html):

- i. derby.jar
 - ii. derbyclient.jar
 - iii. derbynet.jar
 - iv. derbytools.jar
 - c. From Hibernate (<http://www.hibernate.org/downloads.html>):
 - i. antlr-2.7.6.jar
 - ii. c3p0-0.9.1.jar
 - iii. cglib-2.2.jar
 - iv. commons-collections-3.1.jar
 - v. dom4j-1.6.1.jar
 - vi. ehcache-1.5.0.jar
 - vii. hibernate-jpa-2.0-api-1.0.0.Final.jar
 - viii. hibernate3.jar
 - ix. javassist-3.12.0.GA.jar
 - x. jta-1.1.jar
 - xi. slf4j-api-1.6.1.jar
 - d. From JUnit (<https://github.com/KentBeck/junit/downloads>):
 - i. junit-4.9b1.jar
- 3. Load the project into Eclipse.
- 4. Compile the project.
- 5. Create a JAR file for the ORC:
 - a. In Eclipse right click on the project and choose "Export...".
 - b. Choose "Java/JAR file" from the list and press "Next >".
 - c. Check the project on the left side and uncheck everything on the right side of "Select the resources to export".
 - d. Be sure that only "Export all output folders for checked projects" and "Compress the contents of the JAR file" are checked.
 - e. Select the export destination for the JAR file.
 - f. Press "Finish".
- 6. Copy all the required libraries from the `lib` folder and the JAR file of the ORC to the `lib` folder in Axis (`tomcat_folder/webapps/axis/WEB-INF/lib`).
- 7. On Linux: For automated starting of the ORC when the system starts, copy the init scripts from the `init` folder of the ORC project folder to `/etc/init.d`. Edit the paths to Java, Ant, Tomcat and the Ant script to

start the database (`ORCDatabase/rsc/buildDB.xml` in the project folder) at the top of the scripts.

```
Then execute the two commands:  
update-rc.d ORC_Services defaults  
update-rc.d ORC_Database defaults
```

3.1 Installation of Management Interface and BPEL Engine

In order to enable a detailed monitoring of the services provided by the ORC, the ORC has to be instrumented with some monitoring extensions as documented in (<https://sourceforge.net/apps/trac/sla-at-soi/wiki/OrcInstrumentation>)

4 Configuration

The ORC requires configuration after a manual installation. Additionally, the ORC offers two web services, which enable reconfiguration of the ORC during runtime.

4.1 Manual Configuration

1. To be done on the system running the basic services:
 - a. The web services are configured to run on *ORCBasicServices*, therefore it is necessary to edit the `/etc/hosts` file on that system. It is necessary to map the name `ORCBasicServices` to the network interface IP address (not loopback, i.e. `127.0.0.1`) of that machine. In our example:

```
141.21.4.81 ORCBasicServices
```

To make sure if `ORCBasicServices` has been mapped into `141.21.4.81`, one could use the command `ping ORCBasicServices` for testing.

- b. To access the database it is also necessary to edit the `/etc/hosts` file of the system. The IP address of the database server has to be mapped to the name `ORCDatabase`. In our example:

```
141.21.4.108 ORCDatabase
```

Note that the ORC basic services (implemented in Axis 1.4) inside the deployable package is already integrated (deployed) in the tomcat server. If redeploying some service is required for testing purpose, follow the instructions below:

2. Automated starting of the ORC on Linux: This section describes how to start the Database, the BasicServices and the CompositeService automatically when the underlying virtual machine is started. If automated start on boot is to work, it is necessary to boot the virtual machines in the correct order: first the database, then the BasicServices, and finally the CompositeService. Wrong order may lead to problems and unavailable services.

- a. On the system running the ORCBasicServices:

Add the init script `/etc/init.d/ORC_Services` to the right runlevels to add the startup script to the boot process of the machine.

On Ubuntu you can run the following command to do this.

```
update-rc.d ORC_Services defaults
```

- b. On the system running the ORCDatabase:

Add the init script `/etc/init.d/ORC_Database` to the right runlevels to add the startup script to the boot process of the machine.

On Ubuntu you can run the following command to do this:

```
update-rc.d ORC_Database defaults
```

Parts of the ORC are now registered as services and are started automatically at boot time. The services can also be started by hand to test them using the command

```
/etc/init.d/<ServiceName> start.
```

Where `<ServiceName>` = `ORC_Database`, `ORC_BasicServices`, or `ORC_CompositeService`.

4.1.1 Start the ORC manually

1. Open the command terminal and change the path to the `rsc` directory in the database folder (`ORCDatabase`). Run the ant target `startderbydatabase` in the `buildDB.xml`:

```
ant -f buildDB.xml startderbydatabase
```

2. Change to the Tomcat directory (ORCServices/tomcat55 in the installation path of step 6a if you installed from ORC_AllInOne.tar.gz) and start the Tomcat server with the following command:

```
bin/startup.sh
```

4.1.2 Manual Deployment and Undeployment of Services

To deploy or undeploy a service Tomcat has to be started.

1. To deploy a service (e.g. card validation service):

```
java -cp bin:lib/axis/axis.jar:lib/axis/commons-discovery-0.2.jar:lib/axis/commons-logging-1.0.4.jar:lib/axis/jaxrpc.jar:lib/axis/log4j-1.2.8.jar:lib/axis/saaj.jar:lib/axis/wsdl4j-1.5.1.jar org.apache.axis.client.AdminClient src/org/cocome/tradingsystem/webservices/axis/cardValidationService/generated/deploy.wsdd
```

2. To undeploy a service (e.g. card validation service):

```
java -cp bin:lib/axis/axis.jar:lib/axis/commons-discovery-0.2.jar:lib/axis/commons-logging-1.0.4.jar:lib/axis/jaxrpc.jar:lib/axis/log4j-1.2.8.jar:lib/axis/saaj.jar:lib/axis/wsdl4j-1.5.1.jar org.apache.axis.client.AdminClient src/org/cocome/tradingsystem/webservices/axis/cardValidationService/generated/undeploy.wsdd
```

4.2 Configuration Services

4.2.1 Configuration Service

The Configuration Service is used to configure the IP addresses to which the hostnames, used by the ORC, are mapped. If you have installed the ORC with the automated installation and all the components are running on one server you won't need to do this configuration as the mapping is already done. Otherwise you can configure the used IP addresses using this web service.

The web service needs to be deployed on the tomcat server.

Then you can run the client:

```
java -cp
lib/axis/jaxrpc.jar:lib/axis/axis.jar:lib/axis/commons-
logging-1.0.4.jar:lib/axis/commons-discovery-
0.2.jar:lib/axis/wsdl4j-
1.5.1.jar:lib/axis/saaj.jar:ORCConfiguration/OrcConfigCli
ent.jar
org.slasoi.orc.configuration.client.ClientOrcConfig
http://localhost:8080/axis/services/OrcConfig?wsdl
```

For every component of the ORC the client asks you for the IP address of the server it is running on. It gives you the name of the component and asks for the IP address. Enter the IP address and press `Enter`. Enter all the IP addresses in this way. Then the web service adds the hostnames and IP addresses to its `hosts` file and you are done.

4.2.2 Bank Configuration Service

The Bank Configuration Service is used to set the flag which tells the `PaymentDebitService` if it should slow down the response.

In order to set the flag, you have to create a new SOAPUI project and enter the following wsdl location:

```
http://<<IP_of_ORCServices>>:8080/axis/services/
BankConfigurationService?wsdl
```

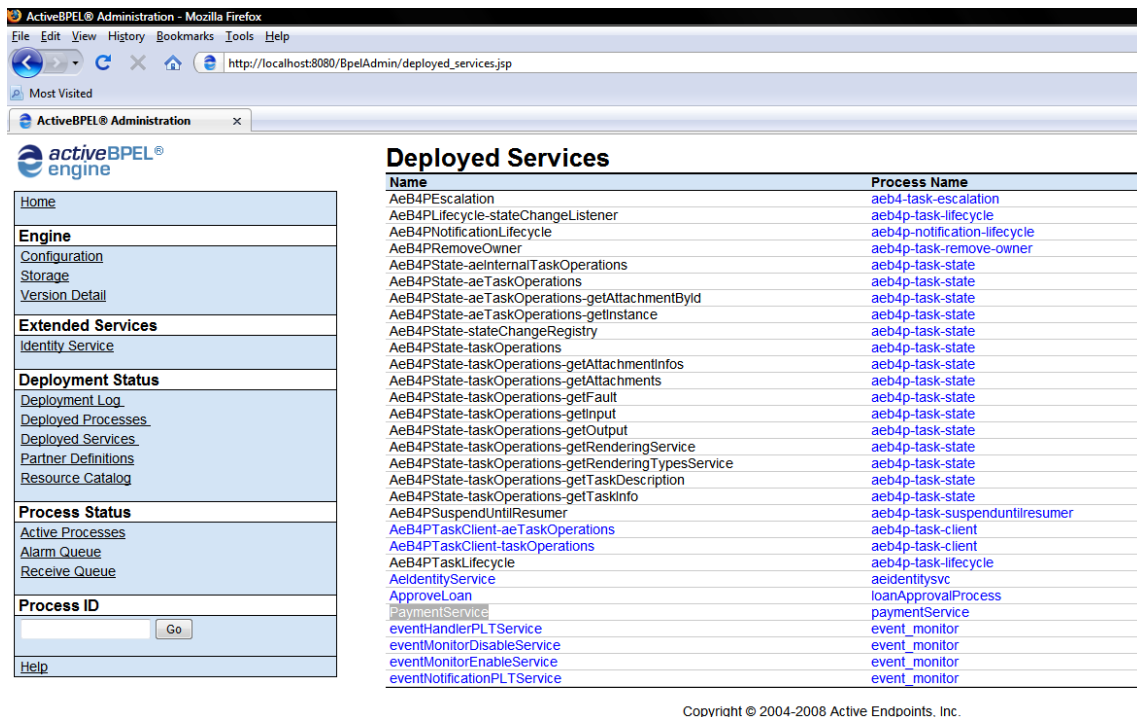
Change the question mark in this line of the request

```
<in0 xsi:type="xsd:boolean">?</in0>
```

into `true` if you want the bank to slow down or `false` if you don't want the response to slow down. Then run the request.

5 Testing the ORC

In order to test the correct deployment of the ORC services you have to open a web browser. Go to the URL `http://<<IP_of_ORCServices>>:8080/axis` and click on `List`. All services of the ORC should be listed including a link to the wsdl. To check the deployment of the composite service `PaymentService`, you have to visit the URL `http://<<IP_of_ORCServices>>:8080/ BpelAdmin/`. Click on `Deployed Services`, the `PaymentService` should be listed there (see Figure 2).



Copyright © 2004-2008 Active Endpoints, Inc.

Figure 2: PaymentService deployed in ActiveBPEL

The functionality of the services can be tested with a SOAP client, such as SOAPUI¹. In the following sections, the tests of the PaymentService and InventoryService are explained.

5.1 PaymentService

In order to test the composite service PaymentService, you have to create a new SOAPUI project and entering following wsdl location:

```
http://<<IP_of_ORCServices>>:8080/active-bpel/
services/PaymentService?wsdl
```

As shown in Figure 3 you have to enter following parameter values:

```
cardInformation: blablabla
cardnumber: 7777
```

¹ <http://www.soapui.org>

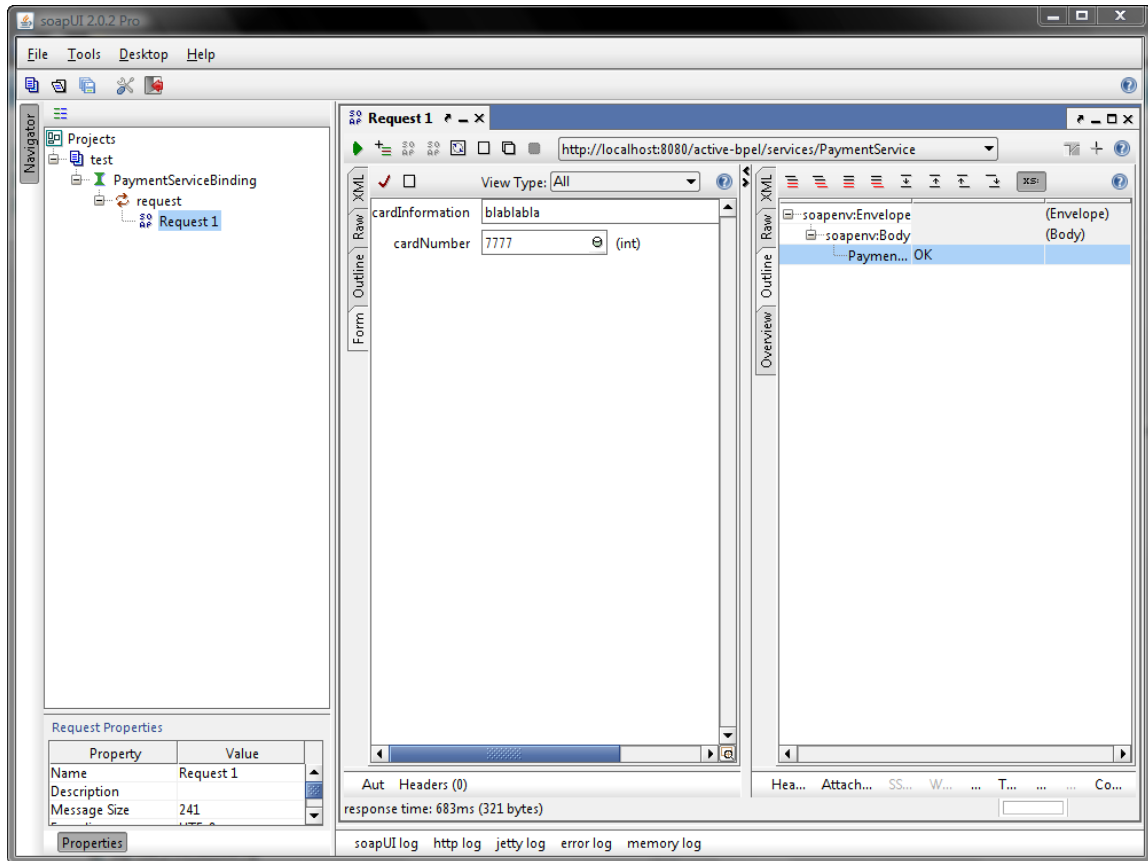


Figure 3: Test PaymentService in SOAPUI

The service should return OK or not enough money.

5.2 InventoryService

Create a new SOAPUI Project with following URL to the wsdl:

<http://<<IP of ORCServices>>:8080/axis/services/InventoryService?wsdl>.

Valid parameter values are:

StoreID: In0 = 65536

BarCode: In1 = 794

5.3 CompositeService

To be done on the system running the CompositeService without the basic services on the same machine.

1. Download deployable ORC Demo – PaymentService (ORCCompositeService.zip) and install/extract it. An example of the installation path is as follows:
2. /usr/ORC_DEMO/
3. When the installation is complete, set the environment variable CATALINA_HOME in the /etc/profile file. An example is as follows:
4. EXPORT CATALINA_HOME=/usr/ORC_DEMO/tomcat
5. Open the command terminal and change the path to bin of the installation path of ORC Demo – PaymentService. Use the command chmod to assign the “execution” right to all the .sh files:
6. chmod a+x *.sh
7. The composition is configured to access services running on *ORCBasicServices*, therefore it is necessary to edit the `hosts` file of the system. This file is located in `/etc`. It is necessary to map the name `ORCBasicServices` to the IP address of the machine running the basic services (See step 3b). In our example:

```
141.21.4.81 ORCBasicServices
```

To make sure if `ORCBasicServices` has been mapped into `141.21.4.81`, one could use the command `ping ORCBasicServices` for testing.

8. Start Tomcat by executing `startup.sh` under the path `bin` of the installation path of ORC Demo – PaymentService. For more information of how to start the ORC Demo on ActiveBPEL, please refer to.