Orchestra User Guide

Orchestra Team ()

- May 2008 -

Copyright © Bull SAS - OW2 Consortium

Table of Contents

Introduction

This documentation is targeted to Orchestra users. It presents the installation procedure and a quick user guide of Nova Orchestra features.

Chapter 1, General information describes the new version Orchestra v4 called Nova Orchestra

Chapter 2, Prerequisites describes the prerequisites to the installation of Nova Orchestra

Chapter 3, Installation guide describes how to install the Orchestra engine

Chapter 4, Configuration and Services describes main configuration features and default services

Chapter 5, User Guide This chapter will guide you through the discovery of the functionalities of Nova Orchestra.

Chapter 1. General information

1.1. Nova Orchestra Overview

Nova Orchestra is the name of new version of Orchestra.

"Nova" technology is based on the "Process Virtual Machine" conceptual model for processes. The Process Virtual Machine defines a generic process engine enabling support for multiple process languages (such BPEL, XPDL…).

On top of that, it leads to a pluggable and embeddable design of process engines that gives modeling freedom to the business analyst. Additionally, it enables the developer to leverage process technology embedded in a Java application.

For more information about the Process Virtual Machine, check Nova Orchestra FAQs [http://wiki.orchestra.objectweb.org/xwiki/bin/view/Main/FAQ] on the Orchestra web site [http://orchestra.objectweb.org].

1.2. Features list

Nova Orchestra is a Web Service Orchestration engine that provides BPEL 2.0 support. Business Process Execution Language (BPEL) is an XML language created by the Oasis Consortium. More information and the specifications can be found on Oasis web site [www.oasis-open.org/committees/wsbpel/]

Nova Orchestra provides partial BPEL 2.0 support. Hereafter, you can find the list of the activities supported :

- Scope

- Receive

- Reply

- Invoke

- Assign

- Empty

- Sequence

- Flow

- If

- Some static analysis

- wait

- while

- repeatUntil

- forEach only in sequential mode

- pick

This version provides Web Service support using the Axis 1.4 framework.

Nova Orchestra is shipped with a complete test suite and a few examples.

Nova Orchestra is persistable. This means that all the data concerning your processes definition and intances execution can be stored in a Database using hibernates. The following database systems have been successfully tested :

- HSQL (default)

- Postgres (8.2)

1.3. Restrictions

Nova Orchestra comes out with an innovative architecture based on a generic and extensible engine, called "The Process Virtual Machine" and a powerful injection technology allowing services pluggability.

This new version of Orchestra is aimed at showing the power of its very innovative architecture by providing support for all the basic activities defined in the BPEL standard. As stated in the previous section, this version provides the possibility to persist the processes definition and execution. Future versions will continue to improve the standard coverage as well as other functionalities like other services. Check the roadmap [http://wiki.orchestra.objectweb.org/xwiki/bin/view/Main/Roadmap] for more information.

This version does not support the following features :

- No fault management

- forEach in parallele mode

- Some restrictions in assign statement :

  - no extensionAssignOperation

  - to partnerlink not supported

  - validate not supported

- Some restrictions in scope statement

  - isolated not supported

  - exitOnStandardFault not supported

  - partnerlink : dynamic update of endpoints

  - handlers not supported

- The following BPEL 2.0 statements are not supported :

  - throw / rethrow

  - exit

  - compensate

  - compensateScope

- validate

- extensionActivity

- import

- extensions

1.4. Tooling

For the new version, Nova Orchestra does not ship a graphical designer. Nova Orchestra engine has been tested with processes created using the Eclipse BPEL designer [www.eclipse.org/bpel/]. Download and installation instruction are available on the project web site.

This version of Nova Orchestra provides an new Web 2.0 administration console. This console will be improved in following releases to add monitoring capabilities.

Chapter 2. Prerequisites

2.1. Hardware

A 1GHz processor is recommended, with a minimum of 512 Mb of RAM. Windows users can avoid swap file adjustments and get improved performance by using 1Gb or more of RAM

2.2. Software

- Nova Orchestra requires Java Development Kit (JDK) 1.5 (also called JDK 5.0) but also runs with with next release.

  The JDK software can be downloaded from http://java.sun.com/j2se/1.5.0

- Nova Orchestra requires Apache Ant 1.7 or higher

  It can be downloaded from http://ant.apache.org

Chapter 3. Installation guide

3.1. Installation

Unzip the orchestra-axis distribution package.

```
>unzip orchestra-4.0.RC.zip
```

A new directory `orchestra-4.0.RC` will be created. It contains an ant file to install and start Orchestra.

Remark : Nova Orchestra runs in Apache Tomcat servlet container. Tomcat 5.5.25 is delivered with the Orchestra Package.

To install Orchestra, go to orchestra directory and launch the install by running ant:

```
>cd orchestra-4.0.RC
>ant install
```

The install script installs Tomcat and Nova Orchestra. The default installation activates the persistence using the HSQL Database.

Important

*if your network is based on a proxy, please change the file `conf/conf.properties`. Set in this file the proxy host, port and non proxy hosts. Then run **ant configure** under the installation directory.*

3.2. Database Management

The default configuration of Nova Orchestra uses the Database persistence service and the HSQL Database. Nova Orchestra has also been tested with the Postgres database system. To change to postgres, you need to put the corresponding JDBC driver in the directory `catalina_base/webapps/orchestra/WEB-INF/lib` to modify the `hibernate.properties` file : uncomment the corresponding lines :

```
# Hibernate configuration

# For using Orchestra with HSQL
# hibernate.dialect                      org.hibernate.dialect.HSQLDialect
# hibernate.connection.driver_class      org.hsqldb.jdbcDriver
# hibernate.connection.url               jdbc:hsqldb:file:db_orchestra
# hibernate.connection.username          sa
# hibernate.connection.password

# For using Orchestra with postgreSQL
# hibernate.dialect                      org.hibernate.dialect.PostgreSQLDialect
# hibernate.connection.driver_class      org.postgresql.Driver
# hibernate.connection.url               jdbc:postgresql://server:port/db
# hibernate.connection.username          user
# hibernate.connection.password          pass

hibernate.dialect                      org.hibernate.dialect.HSQLDialect
hibernate.connection.driver_class      org.hsqldb.jdbcDriver
hibernate.connection.url               jdbc:hsqldb:file:db_orchestra
hibernate.connection.username          sa
hibernate.connection.password

hibernate.hbm2ddl.auto                 update
hibernate.cache.use_second_level_cache true
hibernate.cache.provider_class         org.hibernate.cache.HashtableCacheProvider
hibernate.show_sql                     false
hibernate.format_sql                   false
```

```
hibernate.use_sql_comments             false
hibernate.bytecode.use_reflection_optimizer true
```

3.3. Orchestra directory structure

Hereafter is detailled the structure of Orchestra installation. The installation directory contains the following structure :

```
README
build.xml
install.xml
licence.txt
catalina_base/
catalina_home/
conf/
doc/
examples/
lib/
```

Let's present those items :

- README

  This file gives the basic information related to Nova Orchestra

- build.xml

  This file is an ant file that provides tasks to install and use Nova Orchestra. Just typing `ant` will result giving you the usage.

- install.xml

  This file is an ant file that is called when you run `ant install` (no need to specify the use of this file)

- license.txt

  The license of Nova Orchestra. All of Nova Orchestra is available under the LGPL license.

- conf/

  This directory contains all the configuration files of Nova Orchestra.

  > **Warning**
  >
  > The modifications made to the configuration files are only taken into account after running the `ant configure` command.

- catalina_base/ & catalina_home/

  Those directories are the installation and configuration directories of Tomcat

- doc/

  This directory contains the documentation of Nova Orchestra. It contains 2 directories :

  - html/

    For HTML documentation

  - pdf/

For PDF documentation

- html-single

  For HTML documentation in a single page

- examples/

  This directory contains some examples provided with Nova Orchestra package. The 2 examples provided are :

  - loanApproval

    This is an example from the BPEL 2.0 standards

  - weather

    This example shows how to call a real world Web Service.

- lib/

  This directory contains the libraries used in Nova Orchestra.

Chapter 4. Configuration and Services

This chapter introduces the services configuration infrastructure provided by Nova Orchestra as well as main services included in this version.

4.1. Services Container

The Process Virtual Machine technology includes a services container allowing the injection of services and objets that will be leveraged during the process definition and execution. Objects and services used by the Orchestra engine are defined through a XML file. A dedicated parser and a wiring framework are in charge of creating those objects. Service invoker, publisher, persistence and timers are examples of pluggable services.

This services container (aka IoC container) can be configured through a configuration file. A default configuration file is included in the package under the /conf directory (environment.xml).

4.2. Services

Services in Nova Orchestra is all about pluggability. To allow that, each service has been thought in terms of an interface with different possible implementations. In the following lines you will find a description of main services supported in Nova Orchestra.

The PVM includes a framework to allow the injection of services and objects that will be leveraged during the process definition and execution. Objects and services required in Orchestra are defined through an XML file. A dedicated parser and wiring framework in the PVM is in charge of creating those objects.

A default environment file (environment.xml) is provided in the installed package.

Currently, following objects are required for the execution environment :

- publisher

- invoker

- repository

- persistenceService

- timer

Example of implementation classes for these objects are embedded into the Orchestra jar and defined into the environment.xml file.

4.2.1. publisher

The publisher service sets the way the services proposed by the BPEL processes will be published. The default implementation of this service uses the Axis Web Service Container.

4.2.2. invoker

The invoker service sets the way the BPEL processes will call external services. The default implementation of this service uses the Axis Web Service Container.

4.2.3. Repository

The repository service sets the way the data will be handled by the engine. Nova Orchestra proposes one implementation managing data in the database.

4.2.4. PersistenceService

Persistence is one of key technical services injected into the services container. This service, as well as other major services in Nova Orchestra, is based on a service interface. That means that multiple persistence implementations can be plugged on top.

The Persistence service interface is responsible to save and load objects from a relational database. By default, a persistence implementation based on the Hibernate ORM framework is provided (JPA and JCR to come).

The Process Virtual Machine core definition and execution elements (processes, nodes, transitions, events, actions, variables and executions) as well as the BPEL extension ones (activities, conditions, variables...) are persisted through this service. Process Virtual Machine core elements are also cached by leveraging the default persistence service implementation (Hibernate based). Processes and instances are stored through this persistence service. Repository is the term used in Nova Orchestra to store those entities.

This service is only used if the repository service is set to database.

4.2.5. Timer

The timer service is used for the BPEL statements "wait" and "onAlarm". Dépending on wether Nova Orchestra is running in memory mode or in database mode, the timer service will use a different implementation.

The Database implementation uses the Job Executor module of the Process Virtual Machine for the management of the timers. Its definition in the environment is the following :

For the Job Executor, the administrator can set the number of thread that will manage the jobs. This information is also defined in the environment file with the following line :

```
<job-executor threads='1' auto-start='false' />
```

The default number of thread for the job executoris 1. It is advised to leave this value to avoid concurrency problems.

4.2.6. environment.xml file

The default environment.xml file created during the installation of Nova Orchestra is set to use the database implementation of the persistence service. This file also sets the configuration of hibernate. Here is the environment.xml file generated :

```
<environment-definition>

  <application>
    <hibernate-session-factory table-prefix='NOVA_'>
      <properties resource='hibernate.properties' />
      <mapping resource='hibernate.definition.hbm.xml' />
      <mapping resource='hibernate.execution.hbm.xml' />
      <mapping resource='hibernate.job.hbm.xml' />
      <mapping resource='hibernate.queries.hbm.xml' />
      <mapping resource='hibernate.type.hbm.xml' />
      <mapping resource='hibernate.wire.hbm.xml' />
      <mapping resource='bpel.activities.hbm.xml' />
```

```
        <mapping resource='bpel.elements.hbm.xml' />
        <mapping resource='bpel.execution.hbm.xml' />
        <mapping resource='bpel.runtime.hbm.xml' />
        <mapping resource='bpel.services.hbm.xml' />
        <mapping resource='bpel.queries.hbm.xml' />
        <mapping resource='bpel.util.hbm.xml' />
        <mapping resource='bpel.wsdl.hbm.xml' />
        <mapping resource='bpel.lang.hbm.xml' />
        <cache-configuration resource='pvm.definition.cache.xml' usage='read-write' />
        <cache-configuration resource='bpel.cache.xml' usage='read-write' />
      </hibernate-session-factory>
      <standard-command-service>
        <retry-interceptor />
        <environment-interceptor />
        <transaction-interceptor />
      </standard-command-service>
      <job-executor threads='1' auto-start='false' />
      <repository class='org.ow2.orchestra.execution.services.db.DbRepository' />
      <publisher class='org.ow2.orchestra.axis.AxisPublisher' />
      <invoker class='org.ow2.orchestra.axis.AxisInvoker' />
    </application>

    <block>
      <timer-session />
      <standard-transaction />
      <job-session />
      <hibernate-session />
      <hibernate-bpel-persistence-service />
    </block>

</environment-definition>
```

Chapter 5. User guide

5.1. Start and Stop Nova Orchestra

Nova Orchestra is a webapp that can be deployed on Tomcat. So starting Nova Orchestra in fact starts Tomcat with the correct environment. This can be performed from the installation directory with the following command line :

```
>cd orchestra-4.0.RC
>ant start
```

Starting Orchestra will not be done in background. This means that the console starting Orchestra will be dedicated to the traces from Orchestra. To perform further actions, new consoles need to be opened.

To stop Nova Orchestra, type the following command line :

```
>cd orchestra-4.0.RC
>ant stop
```

5.2. Deploying / undeploying a process

Once Nova Orchestra is started, it is then possible to deploy a new process on the engine :

```
>ant deploy -Dbpel=<process>.bpel -Dwsdl=<process>.wsdl -Dextwsdl=<wsdl1,wsdl2>
```

Nova Orchestra also provides the possibility to use an archive to deploy a process. This archive should be a zip file with the extension .bar. Here is the command line to deploy such an archive :

```
>ant deploy -Dbar=<process>.bar
```

Warning : The archive should be a zip file structured as described bellow :

```
/bar.properties
/<process>.bpel
/<process>.wsdl
/<files>.wsdl
```

The bar.properties file should have the following structure :

```
bpel=<process>.bpel
wsdl=<process>.wsdl
extwsdl=<file1>.wsdl,<file2>.wsdl,...
```

To undeploy a process, use the following command line :

```
>ant undeploy -Dprocess=<process_name>
```

Warning : the process name should be fully qualified. This means that it needs to contain to namespace. For instance :

```
{http://orchestra.ow2.org/weather}weather
```

5.3. Other commands

Nova Orchestra provides a set of other commands that can be usefull

• A command to check the status of Nova Orchestra. This command tells if the engine is started and if so, gives the names of processes deployed on the engine :

```
>ant status
```

- A command to simulate a Web Service call. This command will simulate a WS call to interact with a deployed process :

```
>ant call -Dendpoint=<service_url> -Daction=<SOAP_action> -Dmessage=<message>
```

For example :

```
>ant call -Dendpoint=http://localhost:8080/orchestra/services
          -Daction=http://orchestra.ow2.org/weatherArtifacts/process/weatherPT
          -Dmessage="<weatherRequest xmlns='http://orchestra.ow2.org/weather'>
                    <input>Grenoble,France</input>
                    </weatherRequest>"
```

5.4. Running the examples

The Nova Orchestra package contains two examples of BPEL processes:

- loanApproval : invokes two local web services. This example is taken from the BPEL 2.0 standard.

- weather: invokes a remote Web Service and returns the current weather.

A build.xml file is provided for each of those samples. Those ant scripts provide the same targets to deploy, launch and undeploy the sample. Go to the desired example and use the command lines :

```
>ant deploy
>ant launch
>ant undeploy
```

5.5. Running the tests

Nova Orchestra is delivered with a test suite to check if your installation is correct. There are 3 differents tests available :

- *Core test suite*. This suite tests the core functionnalities of the engine (e.g. BPEL activities, variables, etc...). To run this test suite, the server should not be started.This test suite can be launched with the following command :

```
>ant test
```

- *Axis test suite*. This suite gives the possibility to test the Axis Web Service stack deploying and launching real processes. This test suite can be launched with the following command (server should be started) :

```
>ant test-axis
```

- *Stress test suite*. This suite will launch a small stress test. This test suite can be launched with the following command line :

```
>ant test-stress
```

A command is also provided to launch those 3 test suites at once :

```
>ant test-all
```

The results of the tests are available under the directory `testresults`.

5.6. Configuring Logger

It is possible to activate the logs. To do so, the file `logging.properties` under the directory `$CATALINA_BASE/webapps/orchestra/WEB-INF/classes` can be edited. Here is the content of that file :

```
handlers= java.util.logging.ConsoleHandler
.level= SEVERE
java.util.logging.ConsoleHandler.level = FINEST
java.util.logging.ConsoleHandler.formatter = org.jbpm.util.JbpmFormatter

# For example, set the com.xyz.foo logger to only log SEVERE messages:
# com.xyz.foo.level = SEVERE

#org.jbpm.pvm.level=INFO
#org.ow2.orchestra.level=FINEST
#org.jbpm.pvm.Execution.level=FINEST
#org.jbpm.wire.level=FINEST
```

Uncomment the last lines to activate the logs.