Documentation Bonita

Bonita Team ()

- Dec 2007 -

Table of Contents

Introduction

This documentation is targetted to Bonita users. It presents the installation procedure and a small user guide of Nova Bonita Milestone 2.

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

Chapter 2 , Prerequisites describes the hardware and software prerequisites

Chapter 3, Installation guide describes how to install the Nova Bonita Milestone 2

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

Chapter 5, User Guide guides you through the descovery Nova Bonita functionalities.

Chapter 1. General information

1.1. Nova Bonita introduction

Nova Bonita is the name of new version of Bonita v4.

“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 modelling 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 Bonita FAQs [http://wiki.bonita.objectweb.org/xwiki/bin/view/Main/FAQ] on the Bonita web site [http://bonita.objectweb.org].

1.2. Feature list

Nova Bonita (aka Bonita v4) is a lightweight workflow/BPM solution that provide XPDL support. Nova Bonita M2 adds persistence and standard services to the initial XPDL support provided in the previous milestone. Hereafter you can find the list of features included in this milestone:

- Compliance with the Bonita v3 API (UserBean API with basic operations)
- Support of XPDL 1.0 activities : Join, Split, Manual, Automatic and Route
- Support of XPDL 1.0 elements : Datafield, DataType, Participant, Transition, RedefinableHeader, Transition Restriction and Package
- Support of advanced Bonita v3 entities/resources: Hooks, mappers and performer assignments
- In-memory vs persistent execution
- Default persistence service implementation based on Hibernate
- Core process defintion entities (Process, Node, Transitions...) are cached in the persistent version
- Standard security service based on JAAS LoginModules
- Identity service based on Users, Groups and Memberships. In-memory vs FileSystem based implementations
- Human Task module improvements: users identity, improved Task Repository queries and rolemappers and performer assignments improvements
- Transitions conditions basic support based on BeanShell
- Partial support of ProEd XPDL designer
- Processes, instances, and tasks repositories improvement: in-memory vs persisted implementations
- Variables improvements: project and activity level variable support
- Improvements of unit test coverage

1.3. Restrictions

Nova Bonita Milestone 2 push out an innovative architecture based on a generic and extensible engine, called "The Process Virtual Machine" and a powerful injection technology allowing services pluggability.

Nova Bonita M2 also include basic support for elements defined in the XPDL 1.0 standard. Future milestones will continue to improve the standard coverage and services support, i.e timers, notifications, asynchronous execution... Check the roadmap [http://wiki.bonita.objectweb.org/xwiki/bin/view/Main/Roadmap] for more information.

This milestone does not support the following features:

- Activity variables propagation

  Identity Service: there is no LDAP or ActiveDirectory implementation yet

- block activity
- subprocess
- process versionning
- iteration
- deadline
- role initiator of a process
- Actions are not yet supported (pre-configured hooks based on scripting languages)
- Hooks: processes hook (onInstantiate) as well as activity hooks (before start, after start, onDeadline and OnCancel) are not yet supported
- Role mapper: property and ldap types are not supported
- Performer Assignment: property type is not supported
- Transition conditions based on activity properties (variables)

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 Bonita 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 Bonita requires Apache Ant 1.6.5 or higher

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

Chapter 3. Installation guide

3.1. Installation

Unzip the Bonita distribution package.

```
>unzip bonita-4.0.M2.zip
```

A new directory `bonita-4.0.M2` will be created with the following structure:

```
README
build.xml
License.txt
release_notes.txt
conf/
doc/
examples/
lib/
```

3.2. Bonita directory structure

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

```
README
build.xml
License.txt
release_notes.txt
conf/
doc/
examples/
lib/
```

Let's describe those items :

- README

  This file gives the basic information related to Nova Bonita

- build.xml

  This file is an ant file that provides tasks to run both unit tests and examples (detailled command are given in following sections).

- License.txt

  The license of Nova Bonita. Bonita is released under the LGPL license.

- conf/

  This directory contains the default configuration for Nova Bonita. This xml file, called environment.xml contains the services and objects used as default in Nova Bonita

- doc/

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

  - html/

    For HTML documentation

  - pdf/

    For PDF documentation

- examples/

  This directory contains an example provided with Nova Bonita package. The example is under the directory

  - BonitaSimpleProjectEver

    This is simple Approval Workflow process

- lib/

  This directory contains the libraries used in Nova Bonita.

Chapter 4. Configuration and Services

This chapter introduces the services configuration infraestructure provided by Nova Bonita as well as main services included in this Milestone 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 workflow definition and execution. Objects and services required in Bonita are defined through a XML file. A dedicated parser and wiring framework in the Process Virtual Machine is in charge of creating those objects. Security, identity, persistence, notifications, human task 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 (environment.xml).

Currently, following objects are required for the execution environment:

- deployer object at application context level
- instanceRepository object at application context level
- taskRepository object at application context level
- processRepository object at application context level
- identityService object at application context level
- PersistenceService.NAME is also required at block level

You will find an example of a particular configuration file (environment.xml) under the "conf" directory of Nova Bonita distribution.

4.2. Services

Services in Nova Bonita is all about pluggability. Standard (StandAlone Java based) and Enterprise (J2EE Server based) versions of Nova Bonita can be easily configured thanks to the services container. To allow that, each workflow related 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 Bonita:

4.2.1. Persistence

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

The Persistence service interface (called PersistenceService) is responsible to save and load objects from a relational database. By default, a persistence implementation based on the Hibernate ORM framework (called HibernatePersistenceService) 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 XPDL extension ones (join, split, manual and automatic activities, conditions, variables...) are persisted through this service. Process Virtual

Machine core elements are also cache by leveraging the default persistence service implementation (Hibernate based). Database persistable processes, executions and human task repositories are also based on this persistence service.

4.2.2. Identity

Identity service main objective is to give freedom to system administrators to leverage a particular organization user repository. Traditional user directories such LDAP, ActiveDirectory as well as any other user repository (database or API) can be plugged as implementations of this service.

By default, some user repositories implementations are provided for testing purposes: in memory, basic FileSystem based persistence, and basic database persistence (based on a predefined database schema). Those implementations can also be used in production if there is no other user repository available.

The Identity service is so an extensible interface (known as IdentityServiceOp) build around three main concepts: Users, Groups and Memberships:

- User: a particular user inside an users repository. Users can be created, modified, removed and queried (some of those operations could be not allowed for some repositories (i.e LDAP) through the IdentityService API).
- Group: a group of users in a particular users repository. A group could contain either users security restrictions or hierarchical information. As for users, roles can also be created, removed, modified and queried.
- Membership: a membership represents a user position in a particular group. An user could have two different membership in two different groups. Membership related operations concern set, remove or updates on users position inside groups.

Both Security and Human Task services will leverage the Identity one by checking user login/password and user rights (Security) and by assigning manual activities to users based on some hierarchical information (Human Tasks)

4.2.3. Security

The security service is based on JAAS standard. Main purpose of this service is to provide both authentication and authorization capabilities to the workflow engine. As security directly relates to users permissions, this service also relates to the identity one (commonly security is based on top of the identity service).

As for other services, the Nova Bonita team is concerned on let you the freedom to choose and plug your favorite security implementation. At the same time we also want to provide one ore more default implementations that allow users to quickly set up and start playing with Nova Bonita.

The current implementation of the security service allows to directly leverage default identity service to handle users authentication. Users must login before start calling the Bonita APIs. The current Nova Bonita Milestone is not yet leveraging the Authorization capabilities provided by the security services. In next releases autorization will apply at APIs methods level. We could also imagine fine grained security uses cases in which authorization is not only applied at API method level but also applies to the engine itself, to a particular workflow instance or to the actions executed by the engine.

The Security service is composed by two different JAAS LoginModules. The first one (called IdLoginModule) is responsible to handle security authentication and authorization. The second one (StorageLoginModule) is responsible to keep data of authenticated users. Those login modules can be leveraged in both standard and enterprise environments.

4.2.4. Human Task

Human task management is all about providing the right information to the right people at the right time !. This is one of the most important services that must be provided by a workflow solution.

As human task management can be leveraged in other domains (not only by workflow solutions but by any Java based application) we wanted those features to be a service rather than an internal workflow module. As a result, this service is generic and extensible Task Management service that can be either used in Nova Bonita extension to handle manual task assignments and executions or either by any Java application or Domain Specific Language (i.e BPEL4People extension for instance).

Traditional features such users - roles/group mapping, delegation, scalation, task deadlines handling or manual activities execution life cycle are in scope of this service. Advanced features such configurable activity life cycle, interactions with other task managements system, services or collaborative applications and integration with organizational rules are also part of the main responsabilities of this service.

The current implementation focus on support of manual tasks (also known as manual activities) in Nova Bonita. Basic features such Bonita RoleMappers and Performer Assignments entities allowing users - roles mapping are already supported. Together with the identity and security service, users can login into the system, get their tasks todo list and execute them. All that can be run either in-memory or in a persistent way.

Chapter 5. User guide

This chapter describes how to use Nova Bonita M2. This chapter describe required steps to write a Bonita based sample/application:

- How to create a process definition
- How to import a process definition
- How to develop a simple application by leveraging Bonita APIs

5.1. Designing a xpdl process with ProEd

Like in previous Bonita versions, processes could be created either through a java api or through a graphical editor : ProEd. The java api to build Bonita v4 processes is not yet developped, so processes should be designed under proed (see restrictions below) and then imported as xpdl files. For that, use the stand alone version of the process editor that can be downloaded on Bonita download [http://forge.objectweb.org/project/showfiles.php?group_id=56&release_id=302/] page.

Please, refer to the online [http://wiki.bonita.objectweb.org/xwiki/bin/download/Main/Documentation/Bonita_DevelopmentGuide.pdf] documentation to use proEd.

As final result proEd saves the description of the designed process as an xpdl file that should be imported as described in the following section.

5.2. Getting started with Bonita APIs

If you are already familiar with previous Bonita versions and you have already developped your own applications on top of Bonita, we want to minimize your effort when migrating to Bonita v4. Compatibility from Bonita v3 to v4 is our main concern. For this milestone Bonita has been packaged as a java library and only plain java interfaces are provided (no session beans interface). Bonita v4 Milestone 2 adds a persistence capabilities at both process definition and execution phases. That means that both in-memory or persistent definition and execution can be leveraged by means of configuration.

There is no workflow server implementation yet. However execution concurrency can be achieved by delegating concurrency access to the database. This behaviour can easily be verified by separating the process import operations (XPDLImport POJO) from the execution calls (UserBean POJO). That way, different users can create and handle different workflow instances from a same process. Database lock policy configuration will be responsible for handling concurrency.
<param>Next Nova Bonita milestone will include a server side version in which User and XPDLImport beans will be exposed as SessionBeans. The whole server will run on top of Tomcat (thanks to the integration with an EJB3 lightweight container).</param>

5.2.1. XpdlImport interface

The XPDLImport facade class provides the following static method:

```
void    importDocument(final URL ressourceURL, final Environment environment)
```

5.2.2. User interface

Currently following methods are supported.

- To create an instance of the specified process

```
XpdlExecution instanciateProcess(final String processName)
```

- To obtain all user activities from a particular instance

```
Collection<String> getToDoList(final String instanceName)
```

- To start an activity (when activity state is Ready). its state becomes Executing.

```
void startActivity(final String instanceName, final String nodeName)
```

- To terminate an activity (when activity is executing). Its state becomes Terminated

```
void terminateActivity(String instanceName, String nodeName)
```

5.3. Running the example

The Nova Bonita package contains one example of a XPDL process. This sample is located under examples directory:

- The Bonita Simple Project Ever: a simple Approval Workflow process defintion and execution

The build.xml in the root directory contains required targets to complie and launch the example:

- The Approval Workflow sample can be run in a in-memory environment. That means, no persistence service will be used to store process definition or instances execution related data.

```
>ant memory_examples
```

- Same application (Approval Workflow) can be run using the persistence service. Core Process Virtual Machine entities, XPDL extension as well as execution related data will be persisted in a relational database. By default Nova Bonita embeeds HSQL database and uses it as a default database. You can easily change this default configuration and use your favorite database by modifiying the hibernate.properties file located under "conf" directory.

```
>ant persistence_examples
```

This sample application leverages the Security, Identity and Human Task services so you must provide a right user login/passwd to run the sample. The default identity module (based on HSQL database) is provided with three users ("John", "Jack" and "James" with passwords "cheese", "beer" and "wine" respectively). All three can login into to system but only John and Jack are able to play in the Approval Workflow sample. This behaviour is due to the users - role mapping defined in the previous workflow sample.

The java sample simply import the xpdl file of the process, then creates a workflow instance and leverage getTodoList, startActivity and terminateActivity methods from the User API

Nova Bonita Milestone 2 unit test suite can also be launched from the root directory:

```
>ant tests
```