The BONITA Workflow System
XPDL Support
(Version 1.0)

Miguel Valdés Faura
Marc Blachon

BULL R&D

CHANGES TRACK		
REFERENCES	DATE	CHANGE
1.0		Document creation

INDEX

1 INTRODUCTION

BONITA is a workflow system featuring a lot of innovative features like activities that can start in anticipation, awareness infrastructure allowing users to be notified of any events occurring during the execution in a given process, or automatic activation of user's code according to a defined activity life cycle. Traditional workflow features like dynamic user/roles resolution, activity performer and sequential execution are also included in Bonita to support both cooperative and administrative workflow processes.

Most of those features represent an important added value to the other open source workflow solutions in terms of flexibility, support for cooperatives processes and benefit from application server provided qualities of service. Those features are available to the end user through a set of rich Java/Web Services API's. That way, users are able to define and to execute workflow processes by means of using those APIs.

With the Bonita v2 series, we wanted to standardize the workflow definition process in order to easily allow users to migrate their workflow process definition to Bonita with a minimal cost. That's the reason why we decided to provide support for the XPDL standard. XPDL is the acronym for XML Process Definition Language, an XML based language defined by the WfMC targeting the workflow interoperability between different workflow vendors.
That way we could easily import an existing XPDL process defined with other compliant XPDL workflow vendor in Bonita.
The XPDL support also overcomes the lack of a XML representation of the workflow process definition in the previous versions.

The XPDL support in Bonita has been reached thanks to the Bonita XPDL module having the ability to parse a XPDL document. During the XPDL parsing process, this module will directly call the ProjectSession Bean API as we used to do manually in the previous versions.

If you are not familiar with the XPDL standard, we suggest you to take a look to the XPDL specification document available at http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf

2 EXTENDED ATTRIBUTES

In order to take benefit of the Bonita advanced features using a XPDL based process definition, we have agreed on a couple of advanced attributes which will allow you to use things such Hooks, mappers or performer assignments in Bonita remaining compliant with the XPDL standard. Those attributes are called in XPDL “extended attributes”.
During the next chapters of this document we will describe the extended attributes added by Bonita.

2.1 Role Mapper

A Mapper is unit of work allowing dynamically roles resolution at workflow instantiation time.

Roles in Bonita are mapped with the concept of participants in XPDL. There are different types of performers supported by XPDL but only two of them require a dynamic resolution: “Role” and “Organizational Unit “. That is were mappers are useful.

Possible mapper types for these participant types are:

Participant Type	Possible Mapper Types
Role	Properties, LDAP, Custom
Organizational Unit	Custom

The following table shows an XPDL sample code about what kind of data is required to define mappers using extended attributes:

Possible Mapper Types	XPDL code sample

Ldap	`<Participants>`
	`  <Participant Id="pack_marc01_Par1" Name="Driver">`
	`  <ParticipantType Type="ROLE"/>`
	`  <ExtendedAttributes>`
	`  <ExtendedAttribute Name="Mapper" Value="Ldap"/>`
	`  </ExtendedAttributes>`
	`  </Participant>`
	`</Participants>`

Properties	<Participant Id="Participant_Rep_Par6" Name="VP Sales"> <ParticipantType Type="ROLE"/> <Description>handles sales leads</Description> <ExtendedAttributes> **<ExtendedAttribute Name="Mapper" value="Properties"/>** **<ExtendedAttributes>** </Participant>
Custom	<Participant Id="Participant_Rep_Par6" Name="VP Sales"> <ParticipantType Type="ROLE"/> <Description>handles sales leads</Description> <ExtendedAttributes> **<ExtendedAttribute Name="Mapper" Value="Custom"/>** **<ExtendedAttribute Name="MapperClassName" value="hero.mapper.LdapGroupMembers"/>** **<ExtendedAttributes>** </Participant>

2.2 Performer assignment

A Performer assignment is a unit of work allowing to assign dynamically at execution time the activity performer.

The performer assignment entity is mapped with the concept of performer in XPDL. In fact a performer is the logical performer of a manual activity. In some cases, when the performer need to be determined at execution time the performer assignments could be used. The following table give you an overview about two performer assignment types defined in Bonita.

The performer assignment extended attributes must be defined at XPDL activity level.

Performer assignment Types	XPDL code sample to generate
Property	**<ExtendedAttributes>** **<ExtendedAttribute Name="PerformerAssign" Value="property">** **<Property>valideur</property>** **</ExtendedAttribute/>** **<ExtendedAttributes>**
Callback	**<ExtendedAttributes>** **<ExtendedAttribute Name="PerformerAssign" Value="Callback">** **<Callback>**

	hero.performerAssign.CallbackSelectActors ***<Callback>*** ***<ExtendedAttribute>*** ***<ExtendedAttributes>***

2.3 Hooks

Hooks are user defined logic that can be triggered at some defined points in the life of the activity. Extended attributes are used to meet this Bonita in feature at XPDL activity level.

- Name is : hook
- Value is either the java class name for java hook or a string you want for interactive hook
- Complex content of the extended attribut can include:
- Hook EventName definition (See possible values in Bonita documentation)
- Hook Script

  If a Hook Script is defined then the hook becomes an Interactive Hook.

By default :

- HookEventName is : afterStart (no need to define this event as complex type of the extended attribut).
- Hook type is JAVA for Hook and BSINTERACTIVE for interhook

Ex. of generated XPDL :

```
<ExtendedAttributes>
   <ExtendedAttribute Name="hook" Value="hero.hook.TestHookXpdl">
      <HookEventName>afterStart</HookEventName>
   <HookScript>
     System.out.println("InteractiveBnNodee Hook test, node:"+n.getName());
   </HookScript>
</ExtendedAttribute>
```

2.4 Activity properties

Bonita uses the concept of properties to control the process execution. In Bonita, properties can be defined at two levels :

- the workflow process (this data is defined previously in the document as XPDL datafields or workflow relevant data)
- the activities : acts as a local data during the workflow execution

Extended attributes are used to define properties at activity level because XPDL only support workflow data definition as process level.

In this context, extended attributes related to properties defined at activity level must be included into both datafield and activity levels.

Example of generated XPDL :

```
<DataField Id="choice" Name="choice">
  ...
  <ExtendedAttributes>
    <ExtendedAttribute Name="PropertyActivity" />
  </ExtendedAttributes>
  ...
</DataField>
...
<Activity Id="eval enough holiday" Name="eval enough holiday">
  ...
  <ExtendedAttributes>
    <ExtendedAttribute Name="property" Value="choice"/>
  </ExtendedAttributes>
  ...
</Activity>
```

Activities attributes can be propagated to sub-activities (daughter activities) if the Propagated" tag is added:

```
<Activity Id="eval enough holiday" Name="eval enough holiday">
  ...
  <ExtendedAttributes>
    ...
    <ExtendedAttribute Name="property" Value="choice">
      <Propagated>Yes</Propagated>
    </ExtendedAttribute>
    ...
  </ExtendedAttributes>
  ...
</Activity>
```

2.5 Activities Iteration

Iterations are not explicitly defined in XPDL. In fact XPDL assumes that when a cycle/loop is created in the workflow graph an iteration is defined.

In Bonita we explicitly define iterations as any other workflow entity within the workflow process. Extended attributes are needed in order to support this feature.

Add extended attribute on the activity from which the iteration is expected to start with:

- Name : Iteration
- Value : Condition set on the iteration: this value must be a java condition. In fact this expression will be evaluated at run time by the engine. Within this expression you are able to use the workflow properties as a java variable.
- the destination (To) activity name of the iteration

XPDL code sample generated:

```
<ExtendedAttribute Name="Iteration"
Value="Available_services.equals(&amp;quot;Serv2&amp;quot;)">
    <To>Approval</To>
</ExtendedAttribute>
```

3 IMPORT A XPDL PROCESS IN BONITA

The XPDL import operation has been simplified by including a new task within the build.xml file located under your BONITA_HOME directory.

In fact this task will call a java client which will perform the call to the XPDL Session Bean responsible of the XPDL parsing and the interaction with the Bonita API.

So, as soon as you are ready to import a new XPDL just go to the BONITA_HOME directory and type:

ant import-xpdl -Duser="USER_NAME" -Dpasswd="PASS" -Dxpdl="XPDL_PATH"

In which "USER_NAME" and "PASS" are the credentials of a user having right access to the Bonita workflow engine and "XPDL_PATH" is the path of you XPDL file.