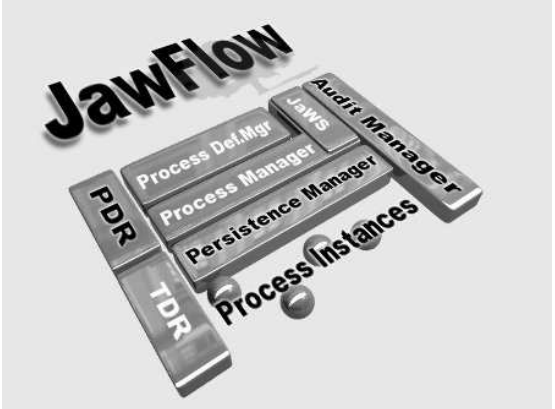
Table of Contents

1. Architecture

JawFlow is composed by many interacting modules. Many of these modules are JMX Mbeans and so they need a JMX Agent to run. To develop, test and run JawFlow we used Jboss AS (www.jboss.org) but since there are no dependencies in the code related to Jboss it can be installed in any application server supporting the JMX technology. We only support here Jboss AS and a detailed installation and configuration procedure is reported in chapter 2



1.1 Process Manager

The ProcessManager is the kernel of JawFlow. It is responsible of process scheduling and enqueueing. It enquires the JaWS module to follow the work path in order to respect the flow depicted in the process definitions.

1.2 Process Definition Manager

The Process Manager uses the Process Definition Manager in order to store, retrieve and delete the process definitions. The process definitions are written in XPDL, the XML Process Definition Language, a standard language proposed by the WfMC coalition. The process definition Manager acts in two ways: the first as a persistence manager for process definitions, the second as a cache manager to avoid the process of loading process definition from the persistence repository. The process definition manager can be plugged into JawFlow respecting some Mbean interfaces. The actual implementation uses Hibernate as a persistence mechanism and a JNDI serialization as a cache manager.

1.3 Persistence Manager

The persistence Manager is in charge of storing and retrieving the Flow status and any change that occurs to it. It stores any information declared as "Extended Attribute" in the Workflow.

1.4 Audit Manager

The Audit Manager is notified of any change that occurs to the status of the activities and of the processes. It is implemented in an asynchronous way using a JMS Queue and a MDB that receives notifications and stores data. The actual implementation of the storage has been done using Hibernate a llowing fast coding of the workflow structure db operations. The Audit Manager is implemented using a chain of delegation pattern, allowing to extend its behaviour by registering other listeners for the change of status notifications.

1.5 JaWS

The Java Workflow System is just a decision engine which, invoked by the ProcessManager, controls the path of the workflow. It is able to recognized decisional structure defined by the WfMC as splits and joins.

1.6 Trigger Manager

It is possible to register a number of "triggers" that, at the occurrence of particular events, can instance a defined process and have it started. The Trigger Manager component controls the triggering subsystem and runs the monitors. Four types of trigger are present in Jawflow:

File Trigger Monitors a directory and its subdirs for any file copied into it

Mail Trigger Monitors a mailbox for e-mails arriving respecting required conditions

Time Trigger It is a sort of "cron" scheduler able to start processes at defined time conditions

FTP Trigger Similar to the File trigger but acts remotely

It is so possible, for instance, to start a particular process when a file is copied into a defined directory or when a mail arrives in a defined mailbox.

1.7 Security Connector

It provides a secure mechanism of talking in a stateless way with the jawflow server. A Standard connector has been written for Jboss but since the methods used are all exposed by the underlying Mbeans it is possible to write your own connector.

1.8 Utility Activities

A number of utility Java activities have been included in the jawflow distributions. For a complete list, please refer to the related documentation.

2. Configuring

jawFlow is provided in bundle with jboss-4.0.4GA Application Server. The distribution should run with a minimum configuration work.

2.1 Database

Jawflow needs a database in order to store process definitions and process status. The database is created via the script jawflow.sql provided with the distribution.

2.2 Configuring Datasources

The jawflow DataSource is defined in the file

server/default/jawflow.deploy/jawflow-ds.xml

and it should be configured pointing to the database created in the previous step.

```
<datasources>
 <local-tx-datasource>
    <jndi-name>jawflowDS</jndi-name>
    <connection-url>...jdbc url...</connection-url>
    <driver-class>...jdbc driver...</driver-class>
    <user-name>...username...</user-name>
    <password>...password...</password>
    <min-pool-size>5</min-pool-size>
    <max-pool-size>20</max-pool-size>
    <idle-timeout-minutes>0</idle-timeout-minutes>
    <track-statements/>
 </local-tx-datasource>
</datasources>
```

The parameters in the previous file are:

jdbc url The JDBC URL of your DB (e.g.: jdbc:postgresql://server/jawflow)

jdbc driver The name of the class implementing the jdbc Driver (e.g.:org.postgresql.Driver)

username a valid username to connect to the database with read/write access

password the user's password

You can of course configure the pooling parameters according to your specific needs.

2.3 Configuring JMS Datasources

jawFlow uses JMS for asynchronous state saving hence it needs a working JMS storage. The default distributions uses a Postgres configured Datasource for the jms Queue but you can use whatever persistent jms mechanism provided by jboss. The provided datasource is in file

server/default/deploy/postgres-ds.xml

and its configuration is similar to the one described previously for the main jawFlow DS. You can of course use the same database for both jawFlow and JMS queue

3. Running

To run jawFlow, just start jboss with the run.sh or run.bat script.

3.1 Checking installation

Once JBoss has started you can check if jawFlow has been started properly too. In order to do this control on the JMX Console:

http://localhost:8080/jmx-console/

if the following JMX Beans have been registered:

```
arsretia.jawflow
 * service=AuditMessageProducer
 * service=DataPersistenceManager
 * service=DataPersister
 * service=DefinitionsStorageManager
 * service=GroupManager
 * service=JNDIStorageManager
 * service=MasterClock
 * service=MonitorMgr
 * service=PersistenceManager
 * service=ProcessCleaner
 * service=ProcessDefinitionRepository
 * service=ProcessManager
 * service=RMIConnectorServer
 * service=SecurityConnector
 * service=TriggerManager
```

3.2 Web User Interface

If everything's ok now, you can access jawFlow's console at the url:

http://localhost:8080/jawflow/Jawflow/home.jsp

and you can reach the logon mask

The default user to access the console is:

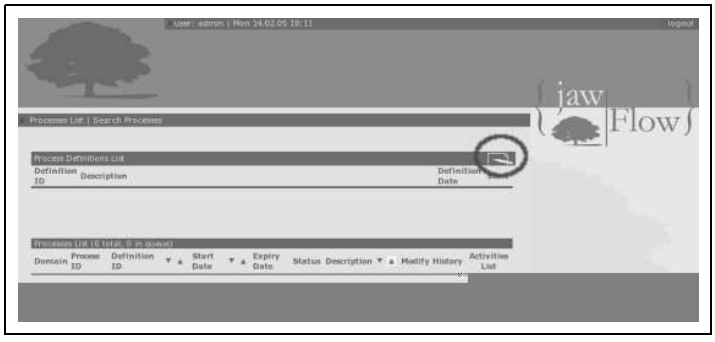
user: admin

password: jawflow

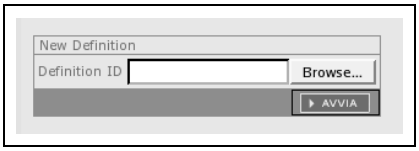
It is strongly suggested to change the password on the db

3.3 Loading a Process definition

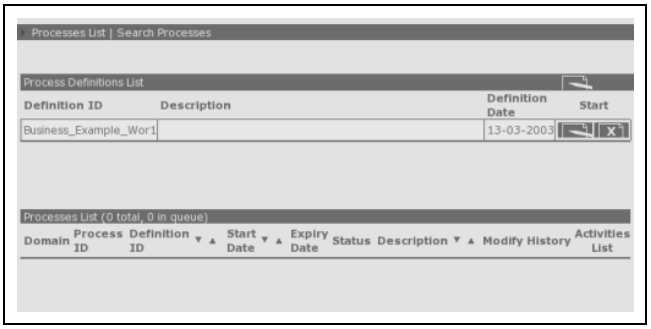
The main jawFlow console is empty at the beginning because we haven't any process definition stored in the database. We can load our XPDL files by clicking on the button circled in red in the following Picture



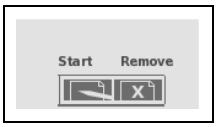
In the following screen, choose your XPDL file clicking on the Browse button and upload your definition with the submit button.



The new definition should appear in the top list.

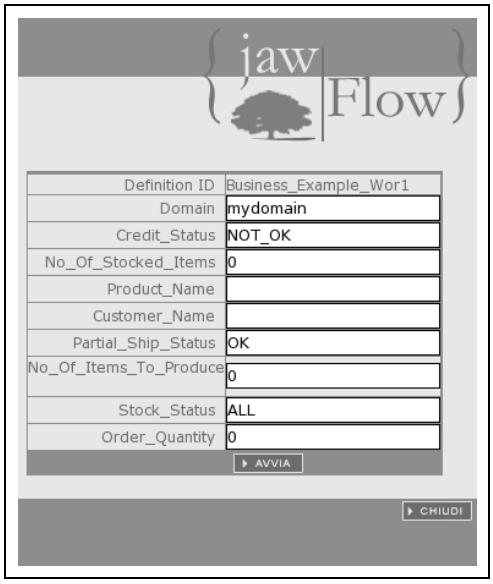


At its right you have the operational button to Start a new Process Instance and to Remove the Definition

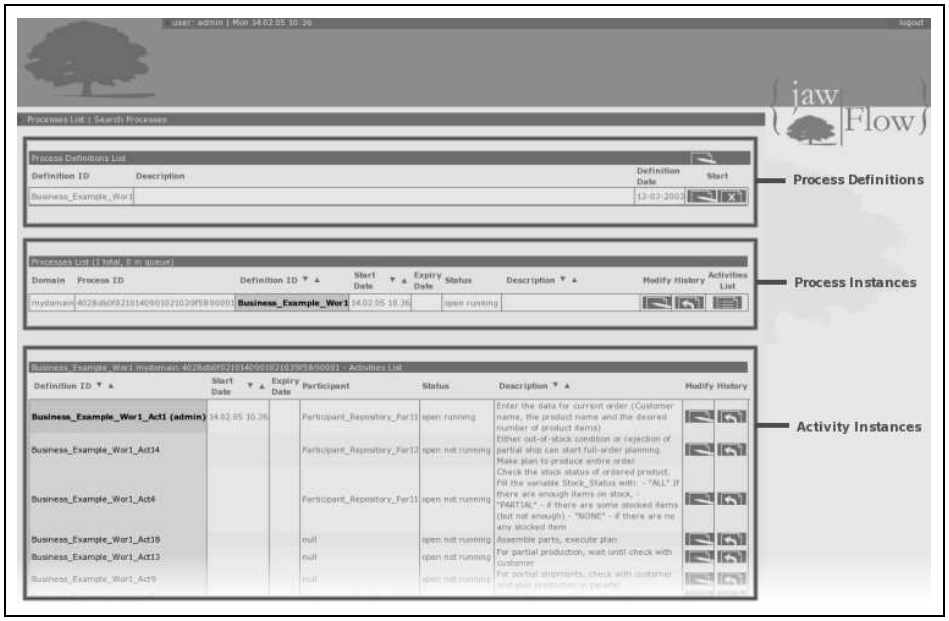


3.4 Starting a Process Instance

To start a New Process Instance, click on the start button. You are presented a start form with the Workflow Relevant Data. Every data defined as a workflow relevant data is presented here and you can change the ones you need.



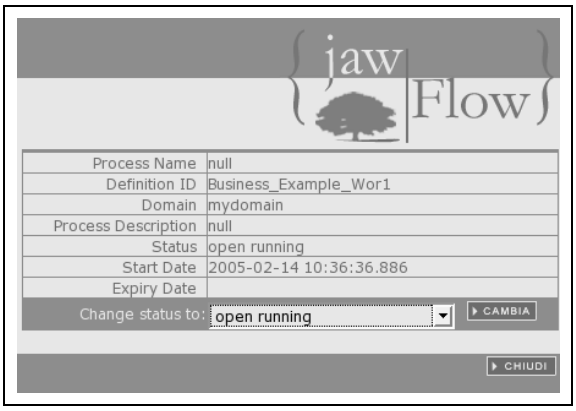
The only mandatory field is the "domain" which represents the JMX domain of your process instance. You can enter a string here, for instance "mydomain", then click on the "start" button.



3.5 Operating on a Process Instance

3.5.1 Changing process status

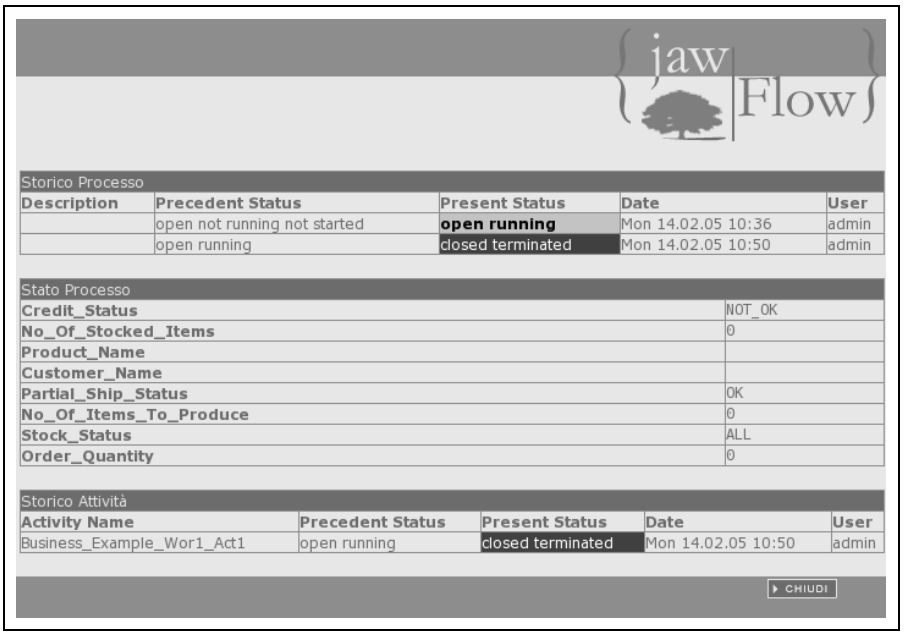
By clicking on the "Modify" button you can change the status of a process instance to a different one



The allowed statuses are the ones defined by the WfMC.

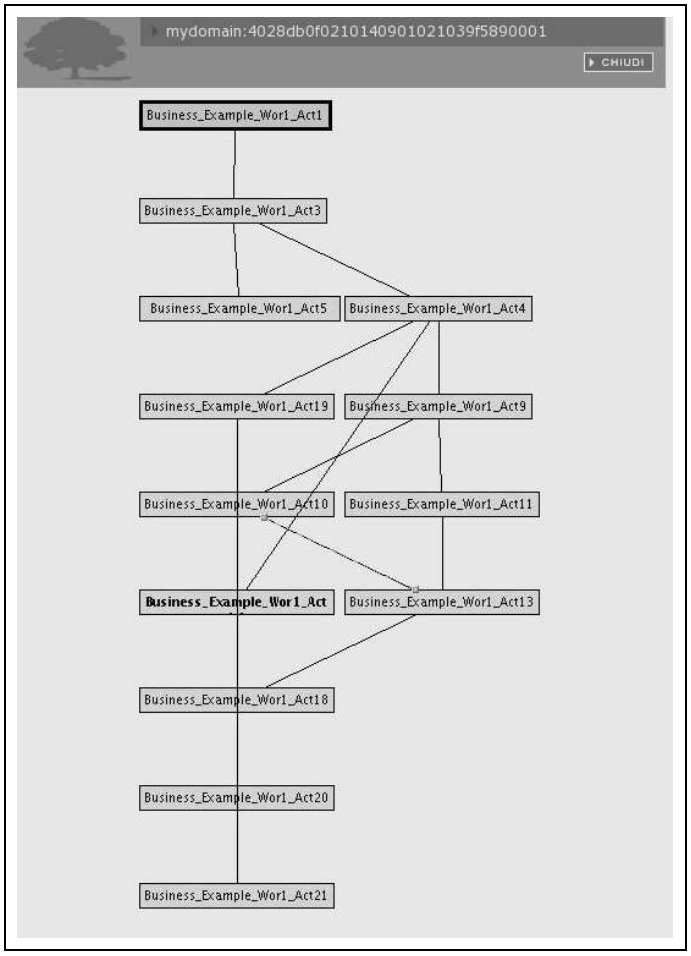
3.5.2 Process Instance History

You can have a look at the process history and at the process internal status.



3.5.3 Process Graph

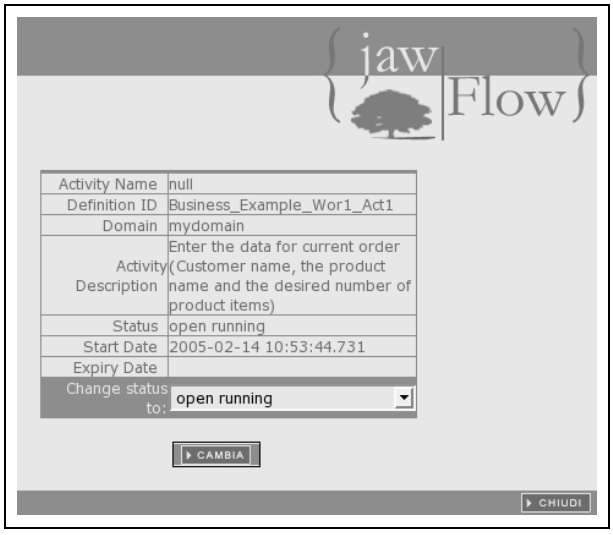
You can have a look at the process graph.



3.6 Operating on an Activity Instance

3.6.1 Changing Activity Instance status

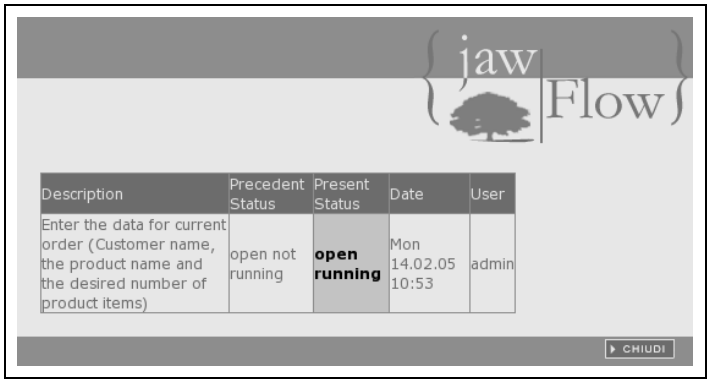
By clicking on the "Modify" button you can change the status of an activity instance to a different one



The allowed statuses are the ones defined by the WfMC.

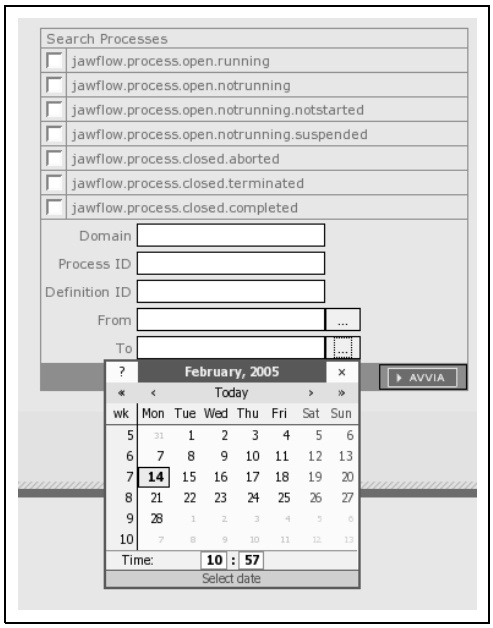
3.6.2 Activity Instance History

You can have a look at the activity history.



3.7 Searching Processes

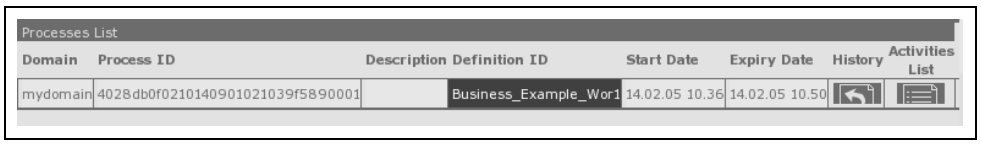
The search utility provides you a way to searc and look at completed processes no more present in the process runtime list (the process runtime list is cleaned of completed processes every now and then)



You can search by:

- Process status
- Process domain
- Process definition id
- Process start and end dates.

The process list you have as a result permits you to have a look at the process history and status and at the process graph.



4. Activities

JawFlow allows you to write activities that are automatically executed by the system. These activities can be written in Java or in a scripting language supported by BSF (using the ScriptActivity)

4.1 Java Activities

The Java Activities can be classes that must implement:

```
net.arsretia.jawflow.kernel.processmgr.IRunnableActivity
```

defined as

```
public
interface IRunnableActivity extends Serializable
{
  public
  int doWork(InputContext oInput,OutputContext oOutput);

  public
  void forceQuit();

  public
```

```
  void gracefulStop();
}
```

The doWork method must return one of the following statuses defined in

```
net.arsretia.jawflow.objectmodel.IStates
```

```
public static final int iACTIVITY_CLOSED             = 0x00F0;
public static final int iACTIVITY_CLOSED_ABORTED     = 0x0020;
public static final int iACTIVITY_CLOSED_TERMINATED  = 0x0040;
public static final int iACTIVITY_CLOSED_COMPLETED   = 0x0080;
```

depending on the end activity status The forceQuit method must stop the activity immediately.
The gracefulStop method can shutdown cleanly the activity.
A typical doWork method can be:

```
public
int doWork(InputContext oInput,OutputContext oOutput);
  while(boRunning){
    // do some important stuff here
  }

  if(boForcedQuit){
    return IStates.iACTIVITY_CLOSED_TERMINATED;
  }

  if(boCleanShutdown){
    return IStates.iACTIVITY_CLOSED_COMPLETED;
  }
  return IStates.iACTIVITY_CLOSED_COMPLETED;
}
```

4.2 Input and Output Context

The doWork method parameters are of the type InputContext and OutputContext In the InputContext there is every parameter declared as

```
<ExtendedAttribute Name="InputVariable" Value="..."/>
```

in the xpdl referring to Workflow Data and every other ExtendedAttribute Since an ExtendedAttribute can refer one or many parameters with the same name the method getAttribute of InputContext always returns a

`java.util.List`

It is responsability of the programmer to fetch the first (and only) attribute in case there is only one.
e.g.:

```
String myVar = (String)((List)oInput.getAttribute("MyVAR")).get(0)
```

The OutputContext allows to set the value of variables declared in the xpdl as

```
<ExtendedAttribute Name="OutputVariable" Value="..."/>
```

If you try to set a variable not declared as OutputAttribute you get an InvalidAttributeException
It is possible, for user defined variables, to refer to a workflow variable with the prefix $
e.g.:

```
<ExtendedAttribute Name="MyVar" Value="$Var1/$Var2"/>
```

4.2 Input and Output Context

The automatic activity is declared in the XPDL with an ExtendedAttribute in the following way:

```
...
<ExtendedAttributes>
  <ExtendedAttribute Name="java.class.name"     Value="<classname>"/>
  <ExtendedAttribute Name="activity.class.path" Value="<classpath>"/>
...
</ExtendedAttributes>
...
```

where

<classname> is the name of the class implementing net.arsretia.jawflow.kernel.processmgr.IRunnableActivity

<classpath> is the classpath where the class can be found

5. The `net.arsretia.jawflow.util` Package

In the `net.arsretia.jawflow.util` package there are many activites ready to be used

5.1 Dos2UnixActivity

This activity converts a DOS file in a UNIX File (CRLF -> CR conversion)

```
<ExtendedAttribute Name="DosFile"        Value="<dosfile>"/>
<ExtendedAttribute Name="DosFileDir"     Value="<dosfiledir>"/>
<ExtendedAttribute Name="UnixFileOutDir" Value="<unixfileoutdir>"/>
<ExtendedAttribute Name="DosFileSuffix"  Value="<dosfilesuffix>"/>
```

```
<ExtendedAttribute Name="UnixFileSuffix" Value="<unixfilesuffix>"/>
<ExtendedAttribute Name="errorVariable"  Value="<errorVariable>"/>
<ExtendedAttribute Name="errorValue"     Value="<errorValue>"/>
```

Var	Description	Mandatory
dosfile	Name of the dos file (without suffix)	
dosfiledir	Directory where the dos file is	
unixfileoutdir	Directory in cui scrivere il file unix	
dosfilesuffix	Dos file Extension	
unixfilesuffix	Unix File Extension (default=".unix")	
errorVariable	Name of the error variable to be set in case of error	
errorValue	Value of the error Variable in case of error	

5.2 Unix2DosActivity

This Activity converts an UNIX file to a DOS one (CR -> CRLF)

```
<ExtendedAttribute Name="UnixFile"        Value="<unixfile>"/>
<ExtendedAttribute Name="UnixFileDir"     Value="<unixfiledir>"/>
<ExtendedAttribute Name="DosFileOutDir"   Value="<dosfileoutdir>"/>
<ExtendedAttribute Name="DosFileSuffix"   Value="<dosfilesuffix>"/>
<ExtendedAttribute Name="UnixFileSuffix"  Value="<unixfilesuffix>"/>
<ExtendedAttribute Name="errorVariable"   Value="<errorVariable>"/>
<ExtendedAttribute Name="errorValue"      Value="<errorValue>"/>
```

Var	Description	Mandatory
unixfile	Unix file name (no suffix)	
unixfiledir	Unix file dir	
dosfileoutdir	DOS File dir	
dosfilesuffix	Dos File Suffix	
unixfilesuffix	Unix File Suffix (default=.unix)	
errorVariable	Name of the error variable to be set in case of error	
errorValue	Valore della variabile di errore in caso di errore	

5.3 ExternalCommandActivity

This activity executes an external command

```
<ExtendedAttribute Name="command"        Value="<command>"/>
<ExtendedAttribute Name="errorVariable"  Value="<errorVariable>"/>
<ExtendedAttribute Name="errorValue"     Value="<errorValue>"/>
```

Var	Description	Mandatory
command	Complete Command	
errorVariable	Name of the error variable to be set in case of error	
errorValue	Valore della variabile di errore in caso di errore	

5.4 FileCopyActivity

File copy

```
<ExtendedAttribute Name="OriginalFile"       Value="<originalfile>"/>
<ExtendedAttribute Name="OriginalFileDir"    Value="<originalfiledir>"/>
<ExtendedAttribute Name="OriginalFileSuffix" Value="<origfilesuffix>"/>
<ExtendedAttribute Name="CopiedFile"         Value="<copiedfile>"/>
<ExtendedAttribute Name="CopiedFileDir"      Value="<copiedfiledir>"/>
<ExtendedAttribute Name="CopiedFileSuffix"   Value="<copiedfilesuffix>"/>
<ExtendedAttribute Name="NameChanges"        Value="<namechanges>"/>
<ExtendedAttribute Name="errorVariable"      Value="<errorVariable>"/>
<ExtendedAttribute Name="errorValue"         Value="<errorValue>"/>
```

Var	Description	Mandatory
originalfile	Name of the original file (without suffix)	
originalfiledir	Directory where the file is	
originalfilesuffix	Original file suffix	
CopiedFile	Copied file name	
CopiedFileDir	Copied file dir	
CopiedFileSuffix	Copied file Suffix	
NameChanges	Can be: LeaveAsItIs no changes ToUpperCase copied file name is uppercase ToLowerCase copied file name is lowercase	
errorVariable	Name of the error variable to be set in case of error	
errorValue	Valore della variabile di errore in caso di errore	

5.5 MailActivity

Sends an email

```
<ExtendedAttribute Name="address"      Value="<address>"/>
<ExtendedAttribute Name="text"         Value="<text>"/>
<ExtendedAttribute Name="smtp"         Value="<smtp>"/>
<ExtendedAttribute Name="from"         Value="<from>"/>
<ExtendedAttribute Name="subject"      Value="<subject>"/>
<ExtendedAttribute Name="attachment"   Value="<attachment>"/>
```

Var	Description	Mandatory
Address	Address (many addresses can be inserted)	
Smtp	SMTP Server	
From	From address	
Text	Text. If the variable is prepended by a '@' is the url of a valid file containing the text	
Subject	Subject	
Attachment	Attachment file url (many attachments can be inserted)	

In the text and the subject it is possible to use the variable substitution using the $ prefix.

5.6 MainWrapperActivity

This activity executes a java class with the method:

```
public static void main(String argv[])
```

```
<ExtendedAttribute Name="runnable.class"      Value="<class>"/>
<ExtendedAttribute Name="runnable.class.path" Value="<classpath>"/>
<ExtendedAttribute Name="runnable.param"      Value="<param>"/>
<ExtendedAttribute Name="errorVariable"       Value="<errorvar>"/>
<ExtendedAttribute Name="errorValue"          Value="<errorval>"/>
```

Var	Description	Mandatory
Class	Class Name	
Classpath	Execution Classpath	
Param	Main class parameters (many parameters can be inserted)	
Errorvar	Name of the variable to be set in case of errors	
Errorval	Value of the variable to be set in case of errors	

5.7 MoveFileActivity

This activity moves a file. It can be configured in the same way as the FileCopyActivity.

5.8 RegexpReplacerActivity

This activity performs a regular expression substitution in a given file

```
<ExtendedAttribute Name="OriginalFile"       Value="<originalfile>"/>
<ExtendedAttribute Name="OriginalFileDir"    Value="<originalfiledir>"/>
<ExtendedAttribute Name="OriginalFileSuffix" Value="<origfilesuffix>"/>
<ExtendedAttribute Name="ReplacedFile"       Value="<repfile>"/>
<ExtendedAttribute Name="ReplacedFileDir"    Value="<repfiledir>"/>
<ExtendedAttribute Name="ReplacedFileSuffix" Value="<repfilesuffix>"/>
<ExtendedAttribute Name="RemoveEmptyLines"   Value="<removeemptylines>"/>
<ExtendedAttribute Name="Regexp"             Value="<regexp>"/>
<ExtendedAttribute Name="Replacestring"      Value="<rreplacestring>"/>
<ExtendedAttribute Name="errorVariable"      Value="<errorvar>"/>
<ExtendedAttribute Name="errorValue"         Value="<errorval>"/>
```

Var	Description	Mandatory
originalfile	Original File Name (without suffix)	
originalfiledir	Original File Dir	
Originalfilesuffix	Original File suffix	
ReplacedFile	Substituted file name	
ReplacedFileDir	Substituted file dir	
ReplacedFileSuffix	Substituted file suffix	
RemoveEmptyLines	It it is true, empty lines are removed	
Regexp	Regular expression	
ReplaceString	String to be inserted	
Errorvar	Name of the variable to be set in case of errors	
Errorval	Value of the variable to be set in case of errors	

5.9 ScriptActivity

This activity executes a script writtine in a language supported by BSF. The Script MUST return one of the codes before mentioned for the Java Activity

```
<ExtendedAttribute Name="language"      Value="<language>"/>
```

```
<ExtendedAttribute Name="script"        Value="<script>"/>
```

Var	Description	Mandatory
Language	Language	
Script	File containing the script	

5.10 UnzipActivity

This Activity unzips a file

```
<ExtendedAttribute Name="ZipInputFile"    Value="<zipinputfile>"/>
<ExtendedAttribute Name="ZipInputFileDir" Value="<zipinputfiledir>"/>
<ExtendedAttribute Name="ExtractDir"      Value="<extractdir>"/>
<ExtendedAttribute Name="PreserveTree"    Value="<preservetree>"/>
<ExtendedAttribute Name="errorVariable"   Value="<errorVariable>"/>
<ExtendedAttribute Name="errorValue"      Value="<errorValue>"/>
```

Var	Description	Mandatory
Zipinputfile	Name of the zip file (without .zip)	
zipinputfiledir	Zip File Dir	
Extractdir	Output Dir	
Preservetree	True if you want to preserve the original dir struct	
errorVariable	Name of the error variable to be set in case of error	
errorValue	Value of the error Variable in case of error	

5.11 ZipActivity

This activity zips a list of files

```
<ExtendedAttribute Name="ZipDir"         Value="<zipDir>"/>
<ExtendedAttribute Name="ZipFileOutDir"  Value="<zipfileoutdir>"/>
<ExtendedAttribute Name="ZipFileName"    Value="<zipfilename>"/>
<ExtendedAttribute Name="Extension"      Value="<extension>"/>
<ExtendedAttribute Name="errorVariable"  Value="<errorVariable>"/>
<ExtendedAttribute Name="errorValue"     Value="<errorValue>"/>
```

Var	Description	Mandatory
ZipfileName	Name of the zip file (without .zip)	
Zipfileoutdir	Zip File Dir	
ZipDir	Directory where the files to be compressed are	
Extension	Extensions of the files to be compressed (many extensions can be inserted)	
errorVariable	Name of the error variable to be set in case of error	
errorValue	Value of the error Variable in case of error	

6 Triggers

JawFlow has an internal mechanism to trigger process instances start when particula events occur. A bunch of triggers are predefined to cover the most common situations.

6.1 TriggerManager

The TriggerManager is the component of the jawFlow architecture responsible for keeping track of all triggers installed in the system and for the effective process spawning,
The TriggerManager can be configured to "listen" on a directory for trigger configuration files. Basically you can install a trigger by copying a configuration file in a given directory and the TriggerManager watch over the directory for any modifications or removal of the file.

6.2 MailTrigger

The MailTrigger monitors an email address and if a mail is received respecting defined rules it starts a given process instance.

```
<trigger>
 <name>MailTrigger</name>
 <version>1</version>
 <class>net.arsretia.jawflow.triggers.mail.MailTrigger</class>
 <processid>
   <rule regexp="Process1*" field="Subject">Process1</rule>
   <rule regexp="Process2*" field="Subject">Process2</rule>
   ...
 </processid>
 <period>5000</period>
 <domain name="domain">
   <var name="var1" value="value1"/>
   <var name="var2" value="value2"/>
   ...
 </domain>
 <host>[mail host]</host>
 <user>[mail username]</user>
 <password>[mail password]</password>
 <protocol>[mail protocol]</protocol>
 <port>[mail host port]</port>
 <maxmessages>1</maxmessages>
 <attachmentdir>[mail attachments dir]</attachmentdir>
```

```
 <debug>false</debug>
</trigger>
```

The processid section includes a list of rules to start different process id if defined rules are met. The "field" attribute can be one of the valid Mail Header file names (e.g.: Subject). The "regexp" attribute is a regexp which is evaluated on the content of the field's value. If the rule is met the processid in the related tag is started. In the above example if the subject starts with "Process1" than a new process instance of th Process1 definition is started.
The other configuration parameters are:

Param	Description	Mandatory
period	Period of the check in ms	
domain	In this section you can add variables to the process instance which is started and you can configure the default domain	
host	Mail Server	
user	Mail User	
password	Mail Password	
protocol	Mail Protocol (supported by javamail)	
port	Mail Port	
maxmessages	Max number of messages to be retrieved each run	
attachmentdir	Directory in which attachment files will be stored	
debug	true to debug javamail	

The started process has the following attributes set:

Param	Description	Type
mail.messagecontents	if message has no attachments it contains message content	java.lang.String
mail.contents	a List of attachment file names	java.util.List
mail.recbcc	a List of BCC addresses	java.util.List
mail.reccc	a List of CC addresses	java.util.List
mail.from	a List of FROM addresses	java.util.List
mail.recto	a List of TO addresses	java.util.List
mail.replyto	a List of REPLYTO addresses	java.util.List
mail.received	date of receival	java.lang.String
mail.sent	date of sending	java.lang.String
mail.attachmentdir	Attachment Dir	java.lang.String
mail.header.*headername*	Value of header *headername*	java.lang.String

6.3 FileTrigger

The FileTrigger monitors a directory tree on the filesystem and if a file is copied/modified it starts a given process instance.

```
<trigger>
 <name>nesteddir</name>
 <version>1</version>
 <class>net.arsretia.jawflow.triggers.file.FileTrigger</class>
 <processid>Test</processid>
 <period>1000</period>
 <rootdir>[rootdir]</rootdir>
 <domain>var1_$var1name1</domain>
 <dir path="dir1" filefilter="*.txt">
   <var name="var1name1" value="var1value1"/>
   <var name="var1name2" value="var1value2"/>
   <var name="var1name3" value="var1value3"/>
   <dir path="dir1.1" filefilter="*.txt1">
    <var name="var1.1name1" value="var1.1value1"/>
    <var name="var1.1name2" value="var1.1value2"/>
   </dir>
   <dir path="dir1.2" filefilter="*.zip">
    <var name="var1.2name1" value="var1.2value1"/>
    <var name="var1.2name2" value="var1.2value2"/>
    <dir path="dir1.2.1" filefilter="*.doc">
     <var name="var1.2.1name1" value="var1.2.1value1"/>
     <var name="var1.2.1name2" value="var1.2.1value2"/>
    </dir>
   </dir>
 </dir>
</trigger>
```

In the above example we are monitoring a tree with the following structure:

```
[rootdir]
    |
    |----dir1
          |
          |----dir1.1
          |
          |----dir1.2
                 |
                 |---dir1.2.1
```

The "filefilter" attribute ensures that only the filtered file are checked, so, for instance, in directory dir1, only .txt files are checked. The started process has the following attributes set:

Param	Description	Type
var	The started process will have this variables as attributes if the file is modified in the corresponding directory	java.lang.String
FileNotificationTimeStamp	Time of notification User	java.lang.String
InputFileDir	directory of modified file	java.lang.String
FileLength	File Lenght	java.lang.Long
LastModified	Last Modified Time	java.lang.String
InputFile	File name without extension	java.lang.String
InputFileSuffix	File Suffix	java.lang.String

6.4 FTPTrigger

The FTPTrigger monitors a file on a remote server and starts a given process instance.

```
<trigger>
 <name>FTPTrigger</name>
 <version>1</version>
 <class>net.arsretia.jawflow.triggers.ftp.FTPTrigger</class>
 <processid>Test</processid>
 <period>5000</period>
 <domain name="domain">
   <var name="var1" value="value1"/>
   <var name="var2" value="value12"/>
 </domain>
 <host>[host]</host>
 <user>[username]</user>
 <password>[password]</password>
 <remotedir>[dir]</remotedir>
 <remotefile>[file]</remotefile>
 <localdir>[localdir]</localdir>
 <mode>[ftp mode]</mode>
 <downloadfile>true</downloadfile>
 <deleteremote>true</deleteremote>
</trigger>
```

The other configuration parameters are:

Param	Description	Mandatory
period	Period of the check in ms	
domain	In this section you can add variables to the process instance which is started and you can configure the default domain	
host	FTP Host	
user	Username	
password	Password	
remotedir	Dir on remote server	
remotefile	File on remote server	
localdir	Local dir where file will be written if downloadfile=true	
mode	FTP Mode	
downloadfile	true if file must be downloaded	
deleteremote	true if file must be deleted after downloading	

6.5 Timerigger

The TimeTrigger acts as a unix crontab starting a process with a scheduler.

```
 <trigger>
  <name>Time</name>
  <version>1</version>
  <class>net.arsretia.jawflow.triggers.time.TimeTrigger</class>
  <processid>Test</processid>
  <period>1000</period>
  <rootdir>[rootdir]</rootdir>
  <domain>var1_$var1name1</domain>
  <crontabentry>[min] [hour] [day of month] [month] [day of week]</crontabentry>
  <crontabentry>[min] [hour] [day of month] [month] [day of week]</crontabentry>
  .....
</trigger>
```

The crontab entry is in the standard unix crontab form (it supports only the wildcard "*")