open your processes

Bonita Open Solution

Version 5.5 Rev 1

Installation Guide

“How to” install BOS on a Tomcat Servlet Container

Change Log

Para 2.5.4 Define variables has been changed.

Contents

Install Bonita Open Solution on a Tomcat Servlet Container

Overview

This document contains instructions for installing Bonita Open Solution (BOS) on a Tomcat web container "from scratch." For installation of a BOS-Tomcat bundle, see the BOS 5.5 Bundle Installation Guide (available for download from the BonitaSoft Web site).

Part 1 gives general requirements for BOS installation.

Part 2 describes how to install BOS on Tomcat starting with the Bonita Open Solution "BOS-5.5-DEPLOY" file.

Part 1. Requirements

1.1 Java requirements

You will need Java Runtime Environment (JRE) version 6 or later. You can also use a Java Development Kit (JDK) version 6 or later.

Bonita Open Solution only supports JRE and JDK provided by Oracle. You can get JRE or JDK from the Oracle web site.

Be sure that JRE or JDK is installed correctly.

If you set up a JRE, remember to set JRE_HOME system variable.
If you set up a JDK, remember to set JAVA_HOME system variable.

To confirm if you are using a JRE or a JDK, check the bin folder. JRE doesn't include javac runtime, JDK does.

1.2 Database requirements

Bonita Open Solution is compatible with any database with an existing Hibernate dialect. Tests have been performed with MySQL, MS SQL, PostGreSQL, and Oracle databases.

Bonita Open Solution also integrates xCMIS to handle files within a process. xCMIS also requires a new, separate database in your DBMS. Supported databases can be found on the eXo web site (look for "dialect").

The following databases should work with BOS:

- MySQL
- PostgreSQL
- Oracle Database
- Microsoft SQL Server
- DB2

Part 2. Install Bonita Open Solution in Tomcat

2.1 Download Tomcat

Download Tomcat 6 binary distribution.

Unzip into a folder and use this path as <TOMCAT_HOME>.

2.2 Download server files

Download the BOS-5.5-deploy.zip (or BOS-SP-5.5-deploy.zip) file from the BonitaSoft Web site (http://www.bonitasoft.com/products/BPM_download.php).

Unzip the file into the folder of your choice.

This location path will be referenced as <BOS-5.5-DEPLOY> in the following sections of this document.

2.3 Set up databases for use with Bonita Open Solution

Create the following databases in your database management system (which must use Hibernate dialect):

- `bonita_journal`
- `bonita_history`

and

- `xcmis`

and associate a user with full rights.

Note that the xcmis database is new with this version of Bonita Open Solution.

2.4 Deploy Bonita in Tomcat

2.4.1 Copy Bonita Execution Engine libraries

Create a folder `<TOMCAT_HOME>\lib\bonita.`

Copy all *.jar files from

```
 <BOS-5.5-DEPLOY>\bonita_execution_engine\engine\libs
<BOS-5.5-DEPLOY>\bonita_execution_enginebonita_client\libs
```

to
`<TOMCAT_HOME>\lib\bonita.`

Modify `<TOMCAT_HOME>\conf\catalina.properties` by adding
`${catalina.home}/lib/bonita/*.jar`
to the property common.loader.

Result:

```
common.loader=${catalina.base}/lib,${catalina.base}/lib/*.jar,
${catalina.home}/lib,${catalina.home}/lib/*.jar,${catalina.hom
e}/lib/bonita/*.jar
```

2.4.2 Deploy User Experience web application (*.war file)

Copy

`<BOS-5.5-DEPLOY> \bonita_user_experience\`
`without_execution_engine_without_client\bonita.war`

to

`<TOMCAT_HOME>\webapps.`

Important for BOS-SP: Remove the `jdtcore*.jar` file.

To do this, open the `bonita.war` file (using an unzip tool) and remove `jdt-core*.jar` from the WEB-INF/lib folder.

2.4.3 Deploy xCMIS web application

Copy `<BOS-5.5-DEPLOY>\xcmis\xcmis.war` to `<TOMCAT_HOME>\webapps`.

2.5 Configure Bonita

2.5.1 Copy Bonita configuration files to web container

Go to `<BOS-5.5-DEPLOY>\conf` and copy the `bonita` folder and `external` folder to `<TOMCAT_HOME>.`

2.5.2 Configure database access

In order to configure Bonita Open Solution to use an external database, you will need to modify some configuration files.

2.5.2.1 Bonita databases

To configure Bonita databases you will need to edit two files:

- `<TOMCAT_HOME>\bonita\server\default\conf\bonita-journal.properties`
- `<TOMCAT_HOME>\bonita\server\default\conf\bonita-history.properties`

Change the default H2 database configuration in each of these files to work with your database.

Default database configuration is defined in journal.properties and history.properties files and it specifies an H2 database. You can comment out H2 configuration with a # at the beginning of the line.

In the "example" section, you will find configuration for most common databases. You can uncomment the desired configuration by removing the #.

You can also use a datasource. In that case, configuration in properties files is lighter and most of the configuration takes place in the Tomcat configuration file: `<TOMCAT_HOME>/conf/context.xml.`

Important: to create a table, enable the option `hibernate.hbm2ddl.auto update`.

For production it's safer to comment out this option *after* table creation (done at the first call to User Experience).

2.5.2.2 xCMIS database

`<TOMCAT_HOME>\external\xcmis\ext-exo-conf` contains two files with database configuration:

```
\exo-configuration-hsql.xml
\exo-configuration-mysql.xml
```

To configure the xCMIS database for use with MySQL, edit the second file above (mysql):

```
   <external-component-plugins>
      <target-
component>org.exoplatform.services.naming.InitialContextInitia
lizer</target-component>
      <component-plugin>
         <name>bind.datasource</name>
         <set-method>addPlugin</set-method>

<type>org.exoplatform.services.naming.BindReferencePlugin</typ
e>
         <init-params>
            <value-param>
               <name>bind-name</name>
               <value>jdbcxcmis</value>
            </value-param>
            <value-param>
               <name>class-name</name>
               <value>javax.sql.DataSource</value>
            </value-param>
            <value-param>
               <name>factory</name>

<value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
            </value-param>
            <properties-param>
               <name>ref-addresses</name>
               <description>ref-addresses</description>
              <property name="driverClassName"
value="com.mysql.jdbc.Driver"/>
              <property name="url"
value="jdbc:mysql://localhost/xcmis"/>
              <property name="username" value="root"/>
              <property name="password" value="root"/>
            </properties-param>
         </init-params>
      </component-plugin>
   </external-component-plugins>
```

Change `localhost/xcmis` to where your xcmis database is located.

Change the `username` and `password` for this database.

NOTE: If you change the database from HSQL, you will also need to change the configuration file that xCMIS uses. Set the variable
`org.exoplatform.container.standalone.config`
to refer to the database it will use (see Define variables, below.)

2.5.2.3 Add database driver

Copy the JDBC driver (`*.jar`) file specific to your database to
`<TOMCAT_HOME>\lib\bonita`.

- MySQL: http://dev.mysql.com/downloads/mirror.php?id=401352#mirrors
- PostgresSQL: http://jdbc.postgresql.org/
- MS SQL: http://msdn.microsoft.com/en-us/sqlserver/aa937724
- Oracle: http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

2.5.3 Configure logging

To configure logging, edit <TOMCAT_HOME>\conf\logging.properties.

See `<TOMCAT_HOME>\external\logging\logging.properties` for a reference configuration. Be sure to define a specific log level for Bonita classes:

```
org.ow2.bonita.level = INFO
org.ow2.bonita.example.level = FINE
org.ow2.bonita.runtime.event.EventDispatcherThread.level =
WARNING

org.hibernate.level = WARNING
net.sf.ehcache.level = SEVERE
```

2.5.4 Define variables

To configure Bonita, system variables need to be defined. To do that, use a setenv script. Two versions of the script, one for Windows and one for Linux, are provided below.

NOTE: If you change the database from HSQL, you will also need to change the configuration file that xCMIS uses.

Uncomment the appropriate line with the variable
`org.exoplatform.container.standalone.config` the script so CMIS_CONFIG points to the database you are using (e.gg MySQL instead of HSQL), and comment out the line that points to hsql.)

Example for Windows: create `<TOMCAT_HOME>\bin\setenv.bat` to set variables.

```
@echo on

rem Sets some variables
set BONITA_HOME="-DBONITA_HOME=%CATALINA_HOME%\bonita"
set LOG_OPTS="-
Djava.util.logging.config.file=%CATALINA_HOME%\external\loggin
g\logging.properties"
set SECURITY_OPTS="-
Djava.security.auth.login.config=%CATALINA_HOME%\external\secu
rity\jaas-standard.cfg"
set CMIS_CONFIG=-
Dexo.data.dir="%CATALINA_HOME%/external/xcmis/ext-exo-data" -
Dorg.exoplatform.container.standalone.config="%CATALINA_HOME%/
external/xcmis/ext-exo-conf/exo-configuration-mysql.xml"
set JAVA_OPTS=%JAVA_OPTS% %LOG_OPTS% %SECURITY_OPTS%
%BONITA_OPTS% %BONITA_HOME% %CMIS_CONFIG% -Dfile.encoding=UTF-
8 -Xshare:auto -Xms512m -Xmx1024m -XX:MaxPermSize=256m -
XX:+HeapDumpOnOutOfMemoryError
```

Example for Linux: create `<TOMCAT_HOME>\bin\setenv.sh` to set variables.

```
#!/bin/sh

# Sets some variables
BONITA_HOME="-DBONITA_HOME=$CATALINA_HOME/bonita"
LOG_OPTS="-
Djava.util.logging.config.file=$CATALINA_HOME/external/logging/logging.properties"
SECURITY_OPTS="-
Djava.security.auth.login.config=$CATALINA_HOME/external/security/jaas-standard.cfg"
CMIS_CONFIG="-Dexo.data.dir=$CATALINA_HOME/external/xcmis/ext-exo-data -
Dorg.exoplatform.container.standalone.config=$CATALINA_HOME/external/xcmis/ext-exo-
conf/exo-configuration-mysql.xml"

CATALINA_OPTS="$CATALINA_OPTS $BONITA_HOME $LOG_OPTS $SECURITY_OPTS
$MEMORY_OPTS $CMIS_CONFIG -Dfile.encoding=UTF-8 -Xshare:auto -Xms512m -
Xmx1024m -XX:MaxPermSize=256m -XX:+HeapDumpOnOutOfMemoryError"
export CATALINA_OPTS
```

2.6 Configure JAAS authentication and communication

Bonita Open Solution comes with a default JAAS (Java Authentication and Authorization Service) configuration. This configuration is provided to the application server by setting the property `java.security.auth.login.config` to JAAS configuration file path.

This configuration comprises two LoginContexts:

- `BonitaAuth` is used by the User Experience when a User tries to log in, to verify that the credentials he provided are correct.
- `BonitaStore` is used every time the User Experience calls the engine, to supply it with the logged-in user's identity.

2.6.1 Configure authentication (BonitaAuth LoginContext)

In the default configuration, `BonitaAuth` consists of only one `LoginModule` :

```
BonitaAuth {
  org.ow2.bonita.identity.auth.BonitaIdentityLoginModule required;
};
```

`BonitaIdentityLoginModule` performs the authentication of the users using the authentication service configured in Bonita environment configuration file:

```
 <authentication-service name='authentication-service'
class='org.ow2.bonita.services.impl.DbAuthentication'>
 <arg><string value='bonita-session:core' /></arg>
 </authentication-service>
```

The default implementation of this service (`DbAuthentication`) goes by the engine database to verify the user credentials.

If you need the authentication to be performed against your own users referential, then you need to provide your own authentication service to replace `DbAuthentication` in the engine environment file. You can also provide your own JAAS `LoginModule` to replace `BonitaIdentityLoginModule` in JAAS configuration file.

To provide your own authentication service, implement the interface
`org.ow2.bonita.services.AuthenticationService`

```
public interface AuthenticationService {
  /**
   * Check whether a user has admin privileges or not
   * @param username the user's username
   * @return true if the user has admin privileges, false otherwise
   * @throws UserNotFoundException
   */
  boolean isUserAdmin(String username) throws UserNotFoundException;

  /**
   * Check some user's credentials
   * @param username the user's username
   * @param password the user's password
   * @return true if the credentials are valid, false otherwise
   */
  boolean checkUserCredentials(String username, String password);
}
```

Note that if you provide your own JAAS LoginModule you still need to implement the authentication service, as the isUserAdmin method will be used to check whether the user is permitted access to the User Experience administration features.

2.6.2 Configure login exchange between Bonita User Experience and Bonita Execution Engine (BonitaStore LoginContext)

In the default configuration, BonitaStore consists of one LoginModule for a deployment in a single application server (Tomcat, JBoss, etc).

In jaas-standard.cfg, BonitaStore is configured to use LocalStorageLoginModule.

If you are using REST, username saved on User Experience side needs to be shared with the Execution Engine. You will need to use a different login module (default is **Local**). Replace LocalStorageLoginModule with BonitaRESTLoginModule in the JAAS configuration file (see Install and Configure REST (optional)).

2.6.3 Configure JAAS for xCMIS

If your xCIMS server is also located here, you must also add the following to java.security.auth.login.config:

```
exo-domain {
 org.exoplatform.services.security.j2ee.TomcatLoginModule
required;
};
```

2.7 Add license file (for BOS-SP)

If you already have a valid SP license, copy the license file to <TOMCAT_HOME>\bonita\server\licenses.

If you need to obtain a valid license, request a license by generating a key. Find the executable file you need in <BOS-5.5-DEPLOY>\security\generateKey-5.5-dist.
If you are not in data source mode, change the following:
When using the Bonita Open Solution web interface (User Experience and Process Web Applications), respect the following patterns for the JAAS login contexts:

2.8 Install and Configure REST (optional)

2.8.1 Deploy REST Web application

Optionally, if you want to use the REST API, specific descriptors and classpath configurations for those servers are included in the bonita-server-rest.war. To obtain this, go to

```
<BOS-5.5-DEPLOY>
\bonita_execution_engine\interfaces\REST\with_engine\bonita-
server-rest.war
```

Open (unzip) this and delete all *.jar files in WEB-INF/lib. Then copy this as a *.war file to <TOMCAT_HOME>\webapps.

2.8.2 Server side

Configure the REST user's login and password in the JAAS configuration to be used by the server to authenticate the client's request for access. This configuration must contain the loginContext `BonitaRESTServer`.

```
BonitaRESTServer {
  org.ow2.bonita.identity.auth.BonitaRESTServerLoginModule
required logins="restuser" passwords="restbpm"
roles="restuser";
};
```

2.8.3 Client side

There are three types of client for the REST server:

1. The Bonita User Experience in REST mode. This requires only configuration.
2. The Bonita API in REST mode (Java client). This requires configuration and acknowledgment on how to use the Bonita API. (See the series Build your applications with Bonita Runtime on www.bonitasoft.org.
3. A language other than Java (HTTP client): This requires acknowledgment on the Bonita REST API.

2.8.3.1 Bonita User Experience configuration

If your client is Bonita User Experience or any Java client:

- Configure the loginContext `BonitaStore`, in the JAAS configuration, according to the REST users defined in the `BonitaRESTLoginModule`. The client will use this login and password defined here to request authentication with the server. For example:

```
BonitaStore {
  org.ow2.bonita.identity.auth.BonitaRESTLoginModule required
restUser="restuser" restPassword="restbpm";
};
```

- Define properties to access the Bonita Execution Engine installed on your server:

```
-Dorg.ow2.bonita.rest-server-address =<your URL> (i.e.
http://localhost:8080/bonita-server-rest/)
-Dorg.ow2.bonita.api-type=REST
-DBONITA_HOME=path/to/bonita/home
```

2.8.3.2 HTTP Client Configuration

In the case where you are directly using an HTTP client, configure HTTP requests based on the Bonita REST API.

All methods, except `checkUserCredentials`, `checkUserCredentialsWithPasswordHash and getIdentityKeyFromTemporaryToken`, will require authentication using HTTP Basic.

For authentication purposes you must use one of the REST users (login and password) defined on the server side. In addition, you must indicate the logged in "final User." This will be used in the process execution. For instance, if you use the REST API to execute a task, it is this "final User" who will be recorded as the Actor who executed the Task.

The "final User" is defined using the form parameter named "options". Use the syntax `user:username`. For instance, if **john** is the final user, set the "options" parameter with the content `user:john`. The options parameter also can be used to `set domain` and `queryList`.

Attention: No verification is done on the server side to check the final user credentials, so you need to manage this. You can use the method `checkUserCredentialsWithPasswordHash` to verify if a given user and password (hashed using SHA-1) are valid.

2.8.3.3 How to use the parameters

The Bonita REST API uses three types of parameters:

- **Path parameter**: path parameters are identified by the symbols "{" and "}" in the URL path. To use them, replace "{parameter name}" by its value. For instance, for the method `getProcessesByState`:

```
/API/queryDefinitionAPI/getProcessesByState/{processState
}
```

  becomes

```
/API/queryDefinitionAPI/getProcessesByState/READY
```

- **Query parameter**: query parameters are placed after the symbol "?" in the URL. They are listed in the method detail. To define a value for a query parameter, put its value after the symbol "=". For instance, for the method `getLightProcessesByIndexAndPageSize`

```
/API/queryDefinitionAPI/getLightProcessesByIndexAndPageSi
ze?fromIndex=�&pageSize=�
```

  becomes

```
/API/queryDefinitionAPI/getLightProcessesByIndexAndPageSi
ze?fromIndex=0&pageSize=20
```

- **Form parameter**: Unlike the query and path parameters, a form parameter is not passed in the URL, but in the body of the HTTP request. All form parameters are listed in the method detail. To use them, add the parameter to the HTTP request content and use the header "Content-Type" with the value "application/x-www-form-urlencoded". For instance, to set john as final user using the "options" parameter (available in all methods), add the header "Content-Type:application/x-www-form-urlencoded" to the HTTP request and add the value "options=user:john" to its content.

2.8.4 Java Object string representation

For examples of how Java objects are mapped on a string, see Example of Java Object String Representation at www.bonitasoft.org.

2.9 Configure multi-tenancy

The Bonita Execution Engine offers full multi tenancy support. Data isolation is established physically: data are isolated in different database schemas. Data from one tenant are isolated in a database schema and are not shared with data from another tenant.

In a multi tenancy implementation the Bonita Execution Engine accepts a different configuration according to each tenant. An engine configuration wraps the database, the repository properties, etc.

It is not necessary to duplicate execution engines. One engine can work with multiple configurations and thus multiple tenants. Each tenant will work with its own database, repository, etc.

2.9.1 Bonita Execution Engine configuration

In `BONITA_HOME/server/`, create a new subfolder for each new tenant. Name the subfolder with the name of the tenant (eg `myTenantName`).

Create a subfolder called `conf` in each new tenant subfolder.

Copy and paste the following files from the default tenant `conf` folder to the new tenant `conf` folder:

- `bonita-server.xml`
- `bonita-journal.properties`
- `bonita-history.properties`

2.9.1.1 Configure `bonita-server.xml` for each tenant

If you are not in data source mode, change the domain, database, large data repository, and xCMIS configurations as shown in the examples below.

Domain:

```
<domain id='myTenantName'/>
```

Database: In the journal and history configurations, change default to myTenantName

```
<hibernate-configuration name='hibernate-configuration:core' >
<properties file='${BONITA_HOME}/server/myTenantName/conf/bonita-
journal.properties' />
...
<hibernate-configuration name='hibernate-configuration:history' >
<properties file='${BONITA_HOME}/server/myTenantName/conf/bonita-
history.properties' />
...
```

Large data repository:

```
<large-data-repository name='large-data-repository'
class='org.ow2.bonita.services.impl.FileLargeDataRepository'>
<arg><string value='${BONITA_HOME}/server/myTenantName/work' /></arg>
</large-data-repository>
```

xCMIS: Be sure to add an implementation specific to each tenant in each section.

```
<!-- Description: Implementation of the documentation manager.
-->
      <documentation-manager name='documentation-manager'
class='org.ow2.bonita.services.impl.CMISDocumentManager'>
        <arg><string value='ATOM' /></arg>
        <arg><string
value='http://localhost:8080/xcmis/rest/cmisatom' /></arg>
        <arg><string value='myTenantName'/></arg>
        <arg>
          <object
class='org.ow2.bonita.services.impl.XCmisUserProvider'>
            <constructor>
              <arg><string value='root'/></arg>
              <arg><string value='exo'/></arg>
            </constructor>
          </object>
        </arg>
      </documentation-manager>
```

2.9.1.2 Configure exo-configuration-<DATABASE>.xml

Go to `BOS-5.5-DEPLOY\conf\external\xcmis\ext-exo-conf\exo-configuration-hsql.xml` and copy and paste the following for each tenant, and then replace default (and description) with the tenant name. If you are using a database other than H2, such as MySQL for example, update the configuration file for that database (eg. `BOS-5.5-DEPLOY\conf\external\xcmis\ext-exo-conf\exo-configuration-mysql.xml`.)

```
<object-param>
               <name>MyTenantName</name>

<description>BonitaMyTenantNameDriver</description>
               <object
type="org.exoplatform.services.cms.drives.DriveData">
                  <field name="name">
                     <string>MyTenantName</string>
                  </field>
                  <field name="repository">
                     <string>db1</string>
                  </field>
                  <field name="workspace">
                     <string>cmis1</string>
                  </field>
                  <field name="permissions">
                     <string>*</string>
                  </field>
                  <field name="homePath">
                     <string>/exo:drives/MyTenantName</string>
                  </field>
                  <field name="icon">
                     <string></string>
                  </field>
                  <field name="views">
                     <string>wcm-category-view</string>
                  </field>
                  <field name="viewPreferences">
                     <boolean>false</boolean>
                  </field>
                  <field name="viewNonDocument">
                     <boolean>true</boolean>
                  </field>
                  <field name="viewSideBar">
                     <boolean>true</boolean>
                  </field>
                  <field name="showHiddenNode">
                     <boolean>false</boolean>
                  </field>
                  <field name="allowCreateFolders">

<string>nt:folder,nt:unstructured</string>
                  </field>
                  <field name="allowNodeTypesOnTree">
                     <string>*</string>
                  </field>
               </object>
            </object-param>
```

Note: In Windows, you will need to escape the "\" (\\) or use "/" in the paths specified.

2.9.1.3 Configure properties files for each tenant

If you are not in data source mode, change the following:

bonita-journal.properties:

```
hibernate.connection.url
jdbc:h2:file:${BONITA_HOME}/server/MyTenantName/work/databases
/bonita_journal.db;FILE_LOCK=NO;MVCC=TRUE;DB_CLOSE_ON_EXIT=TRU
E
```

```
hibernate.search.MyTenantName.indexBase
${BONITA_HOME}/server/MyTenantName/work/indexes/journal
```

bonita-history.properties:

```
hibernate.connection.url
jdbc:h2:file:${BONITA_HOME}/server/MyTenantName/work/databases
/bonita_history.db;FILE_LOCK=NO;MVCC=TRUE;DB_CLOSE_ON_EXIT=TRU
E
```

```
hibernate.search.MyTenantName.indexBase
${BONITA_HOME}/server/MyTenantName/work/indexes/history
```

2.9.1.4 Configure in data source mode in Tomcat

If you are using data source mode in Tomcat, change the following:

bonita-journal.properties:

```
hibernate.connection.datasource
java:/comp/env/bonita/MyTenantName/journal
```

```
hibernate.search.MyTenantName.indexBase
${BONITA_HOME}/server/MyTenantName/work/indexes/journal
```

bonita-history.properties:

```
hibernate.connection.datasource
java:/comp/env/bonita/MyTenantName/history
```

```
hibernate.search.MyTenantName.indexBase
${BONITA_HOME}/server/MyTenantName/work/indexes/history
```

In `<TOMCAT_HOME>/conf/context.xml`, add your data source for journal and history for each tenant:

```
<Resource name="bonita/MyTenantName/journal"
              auth="Container"
              type="javax.sql.DataSource"
              maxActive="100"
              minIdle="10"
              maxWait="10000"
              initialSize="1"
              maxPoolSize="15"
              minPoolSize="3"
              maxConnectionAge="0"
              maxIdleTime="1800"
              maxIdleTimeExcessConnections="120"
              idleConnectionTestPeriod="30"
              acquireIncrement="3"
              testConnectionOnCheckout="true"
              removeAbandoned="true"
              logAbandoned="true"
              username="bonita"
              password="bpm"

              driverClassName="org.h2.Driver"

url="jdbc:h2:file:${BONITA_HOME}/server/MyTenantName/work/data
bases/bonita_journal.db;FILE_LOCK=NO;MVCC=TRUE;DB_CLOSE_ON_EXI
T=TRUE"/>


      <Resource name="bonita/MyTenantName/history"
              auth="Container"
              type="javax.sql.DataSource"
              maxActive="100"
              minIdle="10"
              maxWait="10000"
              initialSize="1"
              maxPoolSize="15"
              minPoolSize="3"
              maxConnectionAge="0"
              maxIdleTime="1800"
              maxIdleTimeExcessConnections="120"
              idleConnectionTestPeriod="30"
              acquireIncrement="3"
              testConnectionOnCheckout="true"
              removeAbandoned="true"
              logAbandoned="true"
              username="bonita"
              password="bpm"
              driverClassName="org.h2.Driver"

url="jdbc:h2:file:${BONITA_HOME}/server/MyTenantName/work/data
bases/bonita_history.db;FILE_LOCK=NO;MVCC=TRUE;DB_CLOSE_ON_EXI
T=TRUE"/>
```

2.9.1.5 Configure in data source mode in JBoss

If you are using data source mode in JBoss, change the following:

`bonita-journal.properties`:

```
hibernate.search.MyTenantName.indexBase
${BONITA_HOME}/server/MyTenantName/work/indexes/journal
```

```
hibernate.connection.datasource
java:bonita/MyTenantName/journal
```

`bonita-history.properties`:

```
hibernate.search.MyTenantName.indexBase
${BONITA_HOME}/server/MyTenantName/work/indexes/history
```

```
hibernate.connection.datasource
java:bonita/MyTenantName/history
```

In `<JBOSS_HOME>/server/default/deploy/<database>-ds.xml`, add one data source for `bonita-journal` and `bonita-history` for each tenant, for example: `<JBOSS_HOME>/server/default/deploy/h2-ds.xml`.

```
<no-tx-datasource>
    <jndi-name>bonita/MyTenantName/journal</jndi-name>
    <connection-
url>jdbc:h2:file:${BONITA_HOME}/server/MyTenantName/work/datab
ases/bonita_journal.db;FILE_LOCK=NO;MVCC=TRUE;DB_CLOSE_ON_EXIT
=TRUE</connection-url>
    <driver-class>org.h2.Driver</driver-class>
    <user-name>sa</user-name>
    <password></password>
    <idle-timeout-minutes>0</idle-timeout-minutes>
  </no-tx-datasource>

  <no-tx-datasource>
    <jndi-name>bonita/MyTenantName/history</jndi-name>
    <connection-
url>jdbc:h2:file:${BONITA_HOME}/server/MyTenantName/work/datab
ases/bonita_history.db;FILE_LOCK=NO;MVCC=TRUE;DB_CLOSE_ON_EXIT
=TRUE</connection-url>
    <driver-class>org.h2.Driver</driver-class>
    <user-name>sa</user-name>
    <password></password>
    <idle-timeout-minutes>0</idle-timeout-minutes>
  </no-tx-datasource>
```

2.9.2 Client configuration

Add the parameter “domain” to the JAAS context login modules and duplicate the login context for each tenant. This will give you the possibility to configure a different authentication mechanism for each tenant. For example:

```
BonitaAuth-tenantId1 {
  org.ow2.bonita.identity.auth.BonitaIdentityLoginModule
required domain=tenantId1;
};
BonitaAuth-tenantId2 {
  org.ow2.bonita.identity.auth.BonitaIdentityLoginModule
required domain=tenantId2;
};

BonitaStore-tenantId1 {
  org.ow2.bonita.identity.auth.LocalStorageLoginModule
required domain=tenantId1;
};
BonitaStore-tenantId2 {
  org.ow2.bonita.identity.auth.LocalStorageLoginModule
required domain=tenantId2;
};
```

When you log into the engine, you will need to specify the tenant you want to log in to by using the right login context.

For example :

```
LoginContext loginContext = new LoginContext("BonitaAuth-
tenantId1",
                           new SimpleCallbackHandler(username,
password);
           loginContext.login();
           loginContext.logout();
           loginContext = new LoginContext("BonitaStore-
tenantId1",
                           new SimpleCallbackHandler(username,
password);
           loginContext.login();
           loginContext.logout();
```

When using the Bonita Open Solution web interface (User Experience and Process Web Applications), respect the following patterns for the JAAS login contexts:

```
BonitaAuth{
  org.ow2.bonita.identity.auth.BonitaIdentityLoginModule
required domain=<tenantId>;
};
BonitaStore{
  org.ow2.bonita.identity.auth.LocalStorageLoginModule
required domain=<tenantId>;
};
```

Warning: There is no java policy to control third-party execution code.

2.10 Deploy

Start your database server and then start Tomcat with `<TOMCAT_HOME>/bin/startup{.bat|.sh}`.

Go to `http://<SERVERADDRESS>:8080/bonita`. Use a FQDN (fully qualified domain name) or IP address for <SERVERADDRESS>.

The Bonita User Experience login page will appear. Log in with administrator user:

- username: admin
- password: bpm

You can also use any of the "default" users: john, jack and james, with the same password.

You can now install a process from Bonita User Experience as an administrator and start a new case of the process. If the case runs without any problem your installation is now operational.

2.11 Potential issues and possible explanations

NOTE: This installation may take several hours to complete.

Behavior	Possible explanation
Unable to create datasource instance	context.xml contains error (property is missing) check your datasource definition (eg Tomcat)
ClassNotFoundException: com.mysql.jdbc.Driver	forgot to add jdbc drivers
Access denied: Connections could not be acquired from the underlying database!	wrong username password specified in context.xml