SpagoBI Installation Manual

Authors Fiscato Luca
Andrea Zoppello

Index

1 Version

Version/Release n° :	0.5	**Data Version/Release :**	Jul, 11th 2005
Update description:	Draft		
Version/Release n° :	1	**Data Version/Release :**	Oct, 20th 2005
Update description:	Final		
Version/Release n° :	1.1	**Data Version/Release :**	Dec, 15th 2005
Update description:	Jboss and JOnAS specification		

2 Document goal

This document provides a detailed description for SpagoBI installation and configuration on a eXo-Tomcat server. SpagoBI can be installed in two different ways:

- using the SpagoBIDemo Installer that simply install a preconfigured SpagoBI portal into a J2EE application server with eXo portal server installed on it. will configure the portal, the application and some examples. The Demo Installer can be downloaded from the SpagoBI repository.
- For a custom installation reading this document and following all the steps described. The document is divided into three parts, the first talk about the installation of the SpagoBI core, the second talk about the eXo portal integration and the third talk about the solution to some commons configuration problems.

3 References

Some of the concepts of this document refer to the following documentation:

- SpagoBI business intellligence platform framework (available at http://spagobi.eng.it/)
- Exo Portal Platform (avaiable at http://www.exoplatform.com)
- Spago framework (avaiable at http://spago.eng.it)

4 Installing SpagoBI Core

For the rest of the document the EXO-HOME name will be used to indicate the file system root directory of spagobi.

4.1 Installing the Metadata Database

SpagoBI metadata is stored in a database (for this release SpagoBI support PostgresSQL, Oracle, and MySQL) . If you don't have anyone of the database servers supported you need to install one of them. Once you have a functional database server you must create a new database for the metadata (spagobi is an example database name suggested).

Once completed the operation above it's possible to proceed with the creation and initial population of the metadata database launching the right script for your database server. For each database server supported you can download from the SpagoBI Repository a zip archive containing the sql script to create the schema, the comments of the table and finally to populate the schema with initial data.
For example, if you have a postgresql database server you need to download the relative archive and run in the following order the scripts contained:

- PG_create.sql
- PG_insert.sql

In every archive exists also a drop script but this one is useful only if you need to clean your database deleting all the spagobi metadata tables.

4.2 Preparing the cms repository

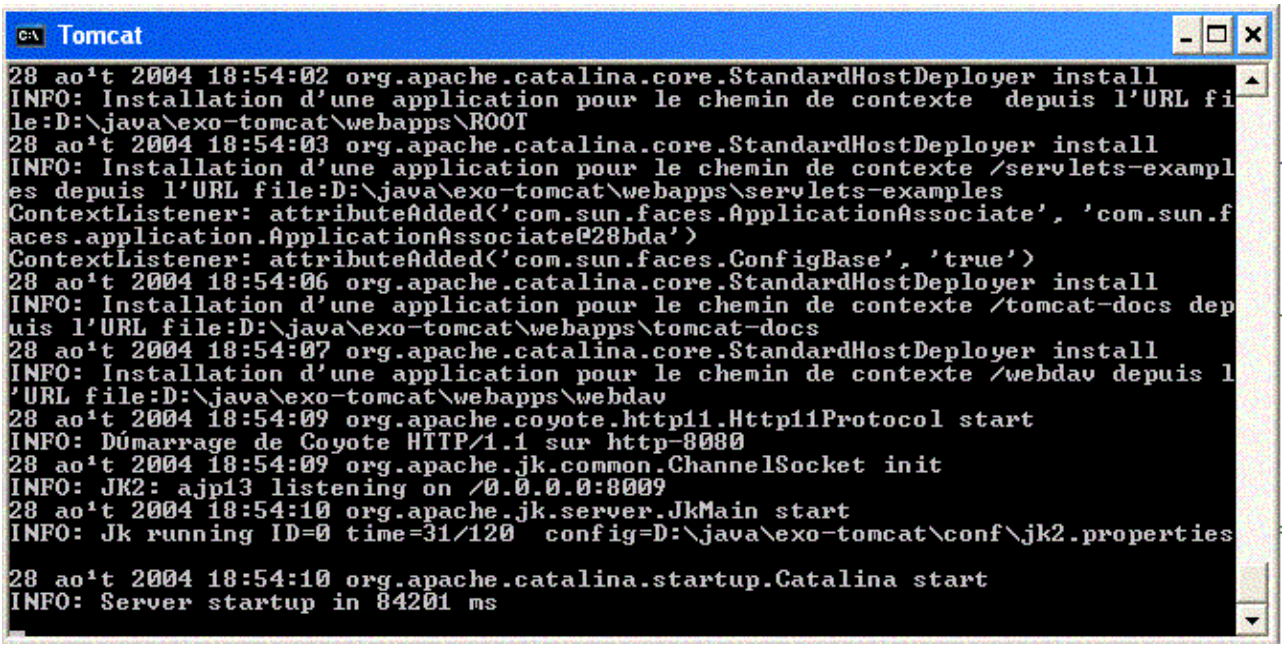
SpagoBI needs a content management system (CMS) for storing the contents that need to be versioned, like the definition of the business object (reports, olap, ...). SpagoBI uses an Apache open source implementation of CMS, called JackRabbit, and it's configured to store the contents into a file system directory. For this reason you must create a new empty folder in your file system (also if distributed) which will be the CMS repository root.

4.3 Install eXo Platform on AS (exo-express ed.)

This part is taken from the exo-platform documentation and reports only for the foundamental steps necessary to install SpagoBI. You can look at the eXo platform documentation for a more complete and exaustive eXo installation and tuning guide.

4.3.1 Install eXo Platform (exo-express ed.) on Tomcat

Download exoplatform express edition 1.0 (exoplatform-tomcat-1.0.zip) from exoplatform site (www.exoplatform.com) . Once the package is downloaded you just have to unzip it in a free choice folder (/pathFolder) and after go to /pathFolder/bin. In the bin directory you can find the startup.bat or the startup.sh script, to run exo on your Windows or unix base system (Linux, Mac OS ...). If all goes well you should have a similar console:



For users which use Windows OS and want to start eXo tomcat as a MS Windows service:

- deploy it in a directory hierarchy that does not contain any spaces.
- do not forget to set the following system properties
  - CATALINA_OPTS=-Dorg.apache.commons.logging.Log="org.apache.commons.logging.impl.SimpleLog $CATALINA_OPTS"
  - CATALINA_OPTS=-Djava.security.auth.login.config="$CATALINA_HOME/conf/jaas.conf $CATALINA_OPTS"

The default distribution comes with the HSQL java database embedded. If you wish to use the eXo Platform in production we recommend you to choose another database. Change the database is quite simple, you just need to change an XML file (that configures the J2EE Datasources) and add the database driver in the classpath of your application server.
The xml file to change is the Tomcat 5 context file of the portal application which is located into the EXO_HOME/conf/Catalina/localhost folder and it's default name is portal.xml. Because user have the possibility to add new portal applications it's necessary to remember that every portal context file must be changed.
Every portal context can define more than one datasource (the default portal defines two datasource, one for the portal metadata and one for workflow) so pay attention to change all your configuration.
For a complete explanation of the context changes to made you can look at the Tomcat 5 documentation in the jndi resources section but, in order to aid you, eXo distribution, for the default portal context, provides several template files like portal.xml.mysql or portal.xml.postgresql. Pick up the one you want and rename it to portal.xml (overide the default one that contains hsql configuration). Of course, you will have to customize your personal database information like username, password and DB URI by editing the file. Finally add the database driver in EXO_HOME/common/lib directory.

4.3.2 Installing eXo platform (enterprise ed.) on Jboss AS 4.01

If we want to use JBoss as J2EE application server, we need to install the enterprise edition of eXo platform.

The enterprise version of exo platform is distributed in a file called **exoplatform-ear-XYZ.zip**.
You will find the **exo.ear** dir and **exo-jboss**, **exo-jonas dir**.

The **exo-jboss** contains the patched files that need to configure your **jboss** like datasource, login-config.xml.

Apply this patch according to the README file contained in exoplatform-ear-XYZ.zip distribution file.

To install the eXo ear file on JBoss you have to:

1. Unzip the exo.ear file as an exploded directory
2. Copy the folder on JBOSS-HOME\server\default\deploy

We need to install exo.ear exploded as a directory because then we're going to deploy spagobi inside the exo.ear archive.

4.3.3 Installing eXo platform (enterprise ed.) on JOnAS AS 4.3.5

If we want to use JOnAS as J2EE application server, we need to install the enterprise edition of eXo platform.

The enterprise version of exo platform is distributed in a file called **exoplatform-ear-XYZ.zip**.
You will find the **exo.ear** dir and **exo-jboss**, **exo-jonas dir**.

The **exo-jonas dir** contains the patched files that need to configure your **jonas** like datasource, login-config.xml.

Apply this patch according to the README file contained in exoplatform-ear-XYZ.zip distribution file.

To install the eXo ear file on JBoss you must:

1. Unzip the exo.ear file as an exploded directory
2. On the unzipped directory, you must unzip as a directory all the wars module of the ear so to have the ear file and all wars modules exploded as directory.
3. Copy the folder on JONAS-HOME\apps\autoload.
4. JOnAS application server resolves recursively all dependent JARs referenced from JAR/WAR modules in "exo.ear", by analyzing "manifest.mf" files. In the original exo distribution case, it fails to load "xercesImpl.jar" referenced by the manifest file in "xalan-2.5.1.jar". After removing it from "xalan-2.5.1.jar" and repackaging this jar there are no more problems. Substitute the reference to the modified jar in all the Manifest.mf file of all the wars module deployed with exo this is because if jonas application server find a

reference to a file that has been replaced by another version an error occurs during EAR deployment.

4.4 Patch eXo to use Hibernate3 into a portlet

For an explanation of why this patch is necessary see the SpagoBI document **Use Hibernate3 in exo-platform 1.0.pdf.**

We need to replace the following libraries:

- asm-1.4.1.jar,
- asm-util-1.4.1.jar
- cglib-full-2.0.1.jar

with:

- asm.jar
- asm-attrs.jar
- cglib-2.1.jar

If we're using exo express edition the library to substitute are in EXO-HOME\common\lib, otherwise if we're using Jboss or Jonas this library are direct contained in the **exo.ear** directory

4.5 Install the database driver

Before to proceed with persistence configuration we must install the database drivers packages in the portal server. Because SpagoBI can be configured to connect to different database server, one for the metadata and one for datawarehouse for example, you must obtain the specific drivers for every database server used by SpagoBI. The drivers package can be obtained from database vendors site and for the current SpagoBI release we test the following versions:

- Postgresql : postgresql-8.0-311.jdbc2.jar
- Oracle: ojdbc14.jar
- MySQL: mysql-connector-java-3.1.10-bin.jar
- HSQLDB: hsqldb-1.8.02.jar

4.6 Install and configure the cms as a JNDI Resource

If we're using exo express edition bundled with tomcat we can configure the CMS Repository as a JNDI resource and so it's necessary to put into the EXO-HOME/commons/lib directory the libraries:

- eng.jackrabbit-0.16.4.1-dev.jar
- eng.jcr.1.0.jar

These libraries can be founded into the directory UtilityFiles/cmsCommonLib of the SpagoBI distribuition and it's not possible to replace them with other versions for classloading problems. SpagoBI uses the Apache Jackrabbit CMS implementation which needs a configuration file for the repository, so it provides one configuration file “repository.xml” into the spagobi war at the /WEB-INF/classes location that you must put into the EXO-HOME/conf directory.
In order to configure the CMS repository like a JNDI resource it's necessary also to put into EXO-HOME/commons/lib directory the library log4j-1.2.8.jar, used by the cms implementation, and to replace the commons-collections-2.1.1.jar library with the commons-collections-3.1.jar.
Finally it's necessary to configure the security permission: edit the jaas.config file into the Exo-HOME/conf directory and add the following instructions:

```
Jackrabbit {
      org.apache.jackrabbit.core.security.SimpleLoginModule required
      anonymousId="anonymous";
};
```

4.7 Install and configure the CMS as a Local SpagoBI Resources

If we’re using exo-platform enterprise edition we must configure the CMS Repository as a local resources owned by SpagoBI application.

In that case we must configure SpagoBI to use Jackrabbit as a local resource. To do this:

1. Edit the EXO.EAR-HOME\spagobi.war\WEB-INF\conf\cms.xml and set the property class on the CONTENT REPOSITORY element to the value:

   it.eng.spago.cms.sessionfactoryimpl.JackrabbitFileSystemImpl.

2. Remove the link to jndi resource in web.xml file, you can do this simple comment the following lines:

```
<resource-ref>
      <description>Spagobi Cms</description>
      <res-ref-name>cms/spagobicms</res-ref-name>
      <res-type>it.eng.javax.jcr.Repository</res-type>
      <res-auth>Container</res-auth>
</resource-ref>
```

3. Edit the file EXO.EAR-HOME\spagobi.war\WEB-INF\conf\jackrabbit.properties and configure it to use a local file system. For example:

```
#jndi_root_name=java:comp/env
#jndi_obj_name=cms/spagobicms


repository_path=C:/SpagoBIJbossCMS
name_configuration_file=repository.xml
```

In that case the jar libraries needed by SpagoBI CMS repository can be placed in EXO-EAR-HOME.

4.8 Configure Datasources

4.8.1 Configuring JNDI Resources in eXo express ed. (Tomcat)

SpagoBI uses the metadata database connection pool and cms repository object as JNDI Resources so you must configure these resources into your eXo platform server. Edit the server.xml file into EXO-HOME/conf directory and add, into the <GlobalNamingResources> tag, the following xml:

```
<!-- For database connection pool -->
<Resource name="jdbc/spagobi" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/spagobi">
      <parameter>
            <name>factory</name>
            <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
      </parameter>
      <parameter>
            <name>driverClassName</name>
            <value>fill with the name of the jdbc drivers</value>
      </parameter>
      <parameter>
            <name>url</name>
            <value>fill with the database connection string</value>
      </parameter>
      <parameter>
            <name>username</name>
            <value>fill with your database user</value>
      </parameter>
      <parameter>
            <name>password</name>
            <value>fill with your database user password</value>
      </parameter>

The following parameters are not mandatory but aid to tuning the connection
pool, it's also possible to add other more specific parameters (look at the
Apache DBCP project for more information)

      <parameter>
            <name>maxActive</name>
            <value>20</value>
      </parameter>
      <parameter>
            <name>maxIdle</name>
            <value>10</value>
      </parameter>
      <parameter>
            <name>maxWait</name>
            <value>-1</value>
      </parameter>
</ResourceParams>
```

```
<!-- For cms Repository -->
<Resource name="cms/spagobicms" auth="Container"
          type="it.eng.javax.jcr.Repository"/>
<ResourceParams name="cms/spagobicms">
      <parameter>
            <name>factory</name>
            <value>org.apache.jackrabbit.core.jndi.BindableRepositoryFactory
            </value>
      </parameter>
      <parameter>
            <name>configFilePath</name>
            <value>fill with the exo platform server root directory path
                  + /conf/repository.xml
            </value>
      </parameter>
      <parameter>
            <name>repHomeDir</name>
            <value>fill with the path of the cms root directory
                    previously created (see point 5.2)
            </value>
      </parameter>
</ResourceParams>
```

When the server starts it will create a Connection pool and Cms repository JNDI resources but they aren't already visible to the SpagoBI application and so you must put into the EXO-HOME/conf directory the following files, which can be founded into the UtilityFiles/exoTomcatContexts directory of the SpagoBI distribuition.

- spagobi.xml
- SpagoBIJasperReportEngine.xml
- SpagoBIJPivotEngine.xml

These files simply define a link to the global JNDI resources for each application.

4.8.2 Configuring Datasources as JNDI Resources in eXo enterprise ed. (Jboss AS 4.01)

To set up a JNDI datasource in JBoss is very simple.

1. First off all ensure you've correctly installed the database driver needed on JBOSS-HOME\server\default\lib.
2. Then for each datasource to configure we need to write a <datasource-name>-ds.xml file and to put in JBOSS-HOME\server\default\deploy folder.

   For example if we want setup two JNDI datasource, one for the metadata and one for the foodmart database we need to write the following files.

Datasource : spagobi-ds.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
<local-tx-datasource>
<jndi-name>spagobi</jndi-name>
<connection-url>jdbc:hsqldb:hsql://localhost/spagobi</connection-url>
<driver-class>org.hsqldb.jdbcDriver</driver-class>
<user-name>sa</user-name>
<password></password>
<min-pool-size>5</min-pool-size>

<!-- The maximum connections in a pool/sub-pool -->
<max-pool-size>20</max-pool-size>
<metadata>
<type-mapping>Hypersonic SQL</type-mapping>
</metadata>
</local-tx-datasource>
</datasources>
```

Datasource : foodmart-ds.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
<local-tx-datasource>
<jndi-name>sbifoodmart</jndi-name>
<connection-url>jdbc:hsqldb:hsql://localhost/foodmart</connection-url>
<driver-class>org.hsqldb.jdbcDriver</driver-class>
<user-name>sa</user-name>
<password></password>
<min-pool-size>5</min-pool-size>

<!-- The maximum connections in a pool/sub-pool -->
<max-pool-size>20</max-pool-size>
<metadata>
<type-mapping>Hypersonic SQL</type-mapping>
</metadata>
</local-tx-datasource>
</datasources>
```

4.8.3 Configuring Datasources as JNDI Resources in eXo enterprise ed. (JOnAS AS 4.3.5)

To setup a JNDI Datasource on JOnAS application server 4.3.5 follow the steps below:

1. Install the database driver on **JONAS-HOME\lib\commons\jonas**
2. For each datasource we need to write a properties file, an to put in JONAS_HOME\conf folder.

   A simple datasource file look like:

   Datasource : spagobi.properties

```
###################### HSQLDB DataSource configuration example
#
#####
# DataSource configuration
# Replace db_jonas and jonas by appropriate values.
#
datasource.name jdbc/spagobi
datasource.url jdbc:hsqldb:hsql://localhost/spagobi
datasource.classname org.hsqldb.jdbcDriver
datasource.username sa
datasource.password
datasource.mapper rdb.hsql
#####
# ConnectionManager configuration
#
# JDBC connection checking level.
# 0 = no special checking
# 1 = check physical connection is still open before reusing it
# 2 = try every connection before reusing it
jdbc.conncheclevel 0



# Max age for jdbc connections
# nb of minutes a connection can be kept in the pool
```

jdbc.connmaxage 1440

Maximum time (in mn) a connection can be left busy.

If the caller has not issued a close() during this time, the connection

will be closed automatically.

jdbc.maxopentime 60

Test statement

jdbc.connteststmt select 1

JDBC Connection Pool size.

Limiting the max pool size avoids errors from database.

jdbc.minconpool 10

jdbc.maxconpool 30

Sampling period for JDBC monitoring :

nb of seconds between 2 measures.

jdbc.samplingperiod 30

Maximum time (in seconds) to wait for a connection in case of shortage.

This may occur only when maxconpool is reached.

jdbc.maxwaittime 5

Maximum of concurrent waiters for a JDBC Connection

This may occur only when maxconpool is reached.

jdbc.maxwaiters 100

3. After datasource definition we need to refer it in the file **JONAS_HOME\conf\jonas.properties** changing the value of the property **jonas.service.dbm.datasources.**

4. At the end you must have in jonas.properties file something like:

   jonas.service.dbm.datasources HSQL1,exo,workflow,ic3,ic3Workflow,sbifoodmart,spagobi

4.9 Install SpagoBI platform on eXo express ed. (Tomcat)

In order to install SpagoBI simply copy the distribution file **spagobi.war** to **EXO-HOME\webapps** folder and start eXo. The SpagoBI application will be exploded. Shutdown the server to continue the SpagoBI core and components configuration.

4.10 Install SpagoBI platform on eXo enterprise ed. (Jboss AS 4.01)

To install SpagoBI platform on eXo platform enterprise edition with jboss application server follow the steps below:

1. Ensure you've correctly installed eXo enterprise edition on Jboss application server like described in 4.4 section of this document.
2. Unzip spagobi.war as an exploded folder and copy in **JBOSS-HOME\server\default\deploy\exo.ear.**
3. All the libraries that are usually located in **\WEB-INF\lib** of the war file must be moved as direct child of the exo.ear directory.
4. After that we must remove from **JBOSS-HOME\server\default\deploy\exo.ear** folder the **commons-collections-3.0.jar** and the **commons-fileupload-1.0.jar** because these two libraries conflict **with collections-3.1.jar** and the **commons-fileupload-1.1-dev.jar**.
5. The manifest file in **JBOSS-HOME\server\default\deploy\exo.ear\spagobi.war\META-INF\MANIFEST.MF** must be created and must refer all the library needed at run-time by SpagoBI platform.
6. Edit the file **JBOSS-HOME\server\default\deploy\exo.ear\META-INF\application.xml** and add a reference to spagobi.war module adding the following lines:

```
<module>
        <web>
                <web-uri>spagobi.war</web-uri>
                <context-root>spagobi</context-root>
        </web>
</module>
```

7. Configure spagobi to use CMS Repository as Local repository like describe in 4.8 section of this document.
8. In Jboss we need to prepare a JBoss web deployment descriptor this, is for introduce a level of indirection between the handling of JNDI resource and the name that this resource haves in web.xml deployment descriptor. In that case with Jboss web deployment descriptor we're able to map a resource defined in web.xml to the one defined in Jboss space. For example:

Jboss Web Deployment Descriptor : jboss-web.xml

```
<jboss-web>
        <resource-ref>
```

```
        <res-ref-name>jdbc/sbifoodmart</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <jndi-name>java:/sbifoodmart</jndi-name>
        </resource-ref>
        <resource-ref>
        <res-ref-name>jdbc/spagobi</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <jndi-name>java:/spagobi</jndi-name>
        </resource-ref>
</jboss-web>
```

4.11 Install SpagoBI platform on eXo enterprise ed. (JOnAS AS 4.3.5)

4.11.1 SpagoBI Installation on JOnAS

1 Ensure you've correctly installed eXo enterprise edition on JOnAS application server like described in 4.5 section of this document.

2 Unzip spagobi.war as an exploded folder and copy in **JONAS-HOME\apps\autoload\exo.ear**

3 All the libaries that are usually located in **\WEB-INF\lib** of the war file must be moved as direct child of the exo.ear directory.

4 After this we must remove from **JONAS -HOME\apps\autoload\exo.ear** folder the **commons-collections-3.0.jar** and the **commons-fileupload-1.0.jar** because these two libraries conflict **with collections-3.1.jar** and the **commons-fileupload-1.1-dev.jar**.

5 Pay attention that in jonas you **must substitute the reference to the modified jar in all the Manifest.mf file of all the wars module deployed with exo** this is because if jonas application server find a reference to a file that has been replaced by another version an error occurs during EAR deployment

6 The manifest file in **JONAS-HOME\apps\autoload\exo.ear\spagobi.war\META-INF\MANIFEST.MF** must be created and must refer all the library needed at run-time by SpagoBI platform.

7 Edit the file **JONAS-HOME\apps\autoload\exo.ear\META-INF\application.xml** and add a reference to spagobi.war module adding the following lines:

```
<module>
        <web>
                <web-uri>spagobi.war</web-uri>
                <context-root>spagobi</context-root>
        </web>
</module>
```

8 Configure spagobi to use CMS Repository as Local repository like describe in 4.8 section of this document.

9 In JOnAS we need to prepare a JOnAS web deployment descriptor this, is for introduce a level of indirection between the handling of JNDI resource and the name that this resource haves in web.xml deployment descriptor. In that case with JOnAS web deployment descriptor we're able to map a resource defined in web.xml to the one defined in JOnAS space. For example:

Jonas Web Deployment Descriptor : jonas-web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<jonas-web-app xmlns="http://www.objectweb.org/jonas/ns"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.objectweb.org/jonas/ns
        http://www.objectweb.org/jonas/ns/jonas-web-app_4_0.xsd" >
        <jonas-resource>
        <res-ref-name>jdbc/spagobi</res-ref-name>
        <jndi-name>jdbc/spagobi</jndi-name>
        </jonas-resource>
        <jonas-resource>
        <res-ref-name>jdbc/sbifoodmart</res-ref-name>
        <jndi-name>jdbc/sbifoodmart</jndi-name>
        </jonas-resource>
</jonas-web-app>
```

4.11.2 Further Operation for JOnAS application server

Due to the different classloading architecture, on JOnAS SpagoBI platform needs to be deployed in a specific way that is different from Jboss and Tomcat.

In particular if we put compiled java files in WEB-INF\classes of a war deployed inside an ear this cause exception of type NoClassDefFound or Class Not Found exceptions

To resolve this problems we'we packaged the **compiled files and spagobi resources** in a jar file called **sbifundamentals.jar** and we've copied at ear level.

The packaging of WEB-INF\classes folder as jar implies that all the configuration on the files that are in that folder must be done before the deploy of spago BI platform (hibernate configuration files, jackrabbit.properties and messages files)

At this point we've had another problems. The **sbifundamentals.jar** contains some classes that extends Spago functionality like **JNDIConnectionPoolFactory.**

This has caused a problems because **JNDIConnectionPoolFactory** is deployed in **sbifundamentals.jar** but is used at runtime by **ConfigServlet** that is a class of Spago that in manifest is referenced before **sbifundamentals.jar**.

To resolve this problem we've splitted the whole file sbifundamentals.jar in two files:

- **spago-ext.jar** (Contains spago classes that are spagobi extension)
- **sbifundamentals.jar** (Contains classes and spagobi resources related to spagobi)

In the manifest file we've referenced the file with the following order:

spago-core-2.0.0.jar spago-ext.jar spago-web-2.0.0.jar sbifundametals.jar

With the above steps all classloading problems have been resolved.

4.12 Configuring SpagoBI

4.12.1 Preliminary Considerations

The following sections are common for all application servers and exo version, the only difference is the location where exo is installed. So in the next sections we'll refer logically to SPAGOBI-HOME to indicate the physical location where SpagoBI is installed.

- In Tomcat with eXo express edition SPAGOBI-HOME will be the folder EXO-TOMCAT-HOME\spagobi

- In Jboss with eXo enterprise edition SPAGOBI-HOME will be the folder JBOSS-HOME\server\default\deploy\exo.ear\spagobi.war\. The EXO-EAR-HOME will be the folder JBOSS-HOME\server\default\deploy\exo.ear\

- In JOnAS with eXo enterprise edition SPAGOBI-HOME will be the folder JONAS-HOME\apps\autoload\exo.ear\spagobi.war\. The EXO-EAR-HOME will be the folder JONAS-HOME\apps\autoload\exo.ear\

4.12.2 Configure the Security Settings

A security provider is used to enable SpagoBI to use the secutity information from the underlying portal server in which SpagoBI is deployed. In particular SpagoBI asks to the secutity provider all the roles available in portal server and the profile of a given user, after authentication.
To install a security provider in SpagoBI two step are necessary:

- copy the provider jar in **\SPAGOBI-HOME\WEB-INF\lib** if you're using exo express edition otherwise copy in **EXO-EAR-HOME** folder if you're using eXo enterprise edition with JBoss or JOnAS.
- configure the provider in **\SPAGOBI-HOME\WEB-INF\conf\spagobi.xml**

SpagoBI distribution contains one provider for eXo Portal which is call **sbi.security.exo.jar** and can be found into the ExoPortalSecurityProvider.zip binary package. If you use eXo-tomcat copy this jar file into the spagobi directory (point 1) and configure the spagobi.xml file (point 2) by adding the following section (or controlling that it exists):

```
<SECURITY>
```

```
	<PORTAL-SECURITY-CLASS>
		it.eng.spagobi.security.ExoPortalSecurityProviderImpl
	</PORTAL-SECURITY-CLASS>
	<USER-PROFILE-FACTORY-CLASS>
		it.eng.spagobi.security.ExoPortalUserProfileFactoryImpl
	</USER-PROFILE-FACTORY-CLASS>
	<ROLE-NAME-PATTERN-FILTER>.*</ROLE-NAME-PATTERN-FILTER>
</SECURITY>
```

SpagoBI imports the roles defined into the portal but, because you probably want to import only a subset of roles, it's possible to filter them based on their names. The filtering rule is a regular expression that can be configured into the <ROLE-NAME-PATTERN-FILTER> tag of the spagobi.xml (see previous xml envelope). Example of regular expression can be ".*" that will match all the roles, or "/spagobi/*" which will match all the roles that start with "/spagobi/" prefix.

4.12.3 Configure SpagoBI persistent layer

In SpagoBI different persistence mechanism are used to store and retrieve data. In a minimal configuration there must be at least:

- a connection with a JSR170 compliant repository.
- a connection with a relational database for metadata persistence layer

Only the second type of connection, for this release, needs some configuration into spagobi core, but before see how to configure it we must make some important consideration.

- Each content repository location is related in unique mode with the metadata database. This means that, for a particular installation, we must never change the connection to the metadata database without changing the content repository location and vice versa.
- For the connection to the relational database SpagoBI use a mix of different technologies, for example for the relational part or the metadata persistence Hibernate is used for detail and the crud operation, but the Spago Data Access Layer is used for the presenting list. This is because hibernate is useful for crud operation but Spago is very useful for list reendering.

For the persistence of the metadata SpagoBI use hibernate 3.05 and spago data-access layer. Only the first one need some additional configuration, infact, based on the type of database server used SpagoBI defines three different hibernate configuration files:

- **hibernate.cfg.xml** (Default): contains the configuration and the reference to the mapping file for postgresql
- **hibernate.cfg.mysql.xml**: contains the configuration and the reference to the mapping file for postgresql
- **hibernate.cfg.ora.xml**: contains the configuration and the reference to the mapping file for postgresql.

To specify wich file to use, edit the **\SPAGOBI-HOME\WEB-INF\conf\spagobi\ spagobi.xml** and, based on your database server, put one of the file names above in **HIBERNATE-CFGFILE** element, for example:

```
<HIBERNATE-CFGFILE>
	hibernate.cfg.mysql.xml
</HIBERNATE-CFGFILE>
```

4.13 Drivers and Engines

SpagoBI is a platform designed to support different products (open source or not) for producing business intelligence artefacts. Those external products in SpagoBI are called **engines**. Examples of engines are Jasper Report, Crystal Clear, Business Object, Jpivot etc. Pratically speaking an engine is a web application that, once invoked with an url request, produces a business intelligence artefacts.
Each single engine has different syntax, parameters, and configuration to produce output but on the other side SpagoBI is a generic platform where business intelligence documents are stored and described (in metadata database and in jcr repository) in generic way. For this reason an integration layer between SpagoBI and each specific engine is needed. The piece of software that implements this integration layer is called **driver** and his function is to build the correct url request, for the engine which it is associated, starting from the data definition into the SpagoBI platform.
SpagoBI platform implements an administration service that allows the administrator users to define one or more logical names for each engine application and associate to each of them the complete class name of the engine driver and the engine url.

4.13.1 Install an engine

A SpagoBI engine is a web application that from a request url produce a business intelligence output artefacts like a report for example. It's possible to connect to some engines produced by other software company (Crystal Clear or Oracle Reports for example) or to produce a new engine web application, like the SpagoBIJasperReport Engine. In both cases in order to install an engine you must deploy it into an application server which may not be the same of the one where SpagoBI is deployed.
After the engine installation you must configure it into the SpagoBI engines service (the administration portlet) which allows you to give a logical name to each engine application and defining the calling url and the driver class name to adapt SpagoBI objects to specification and requirements of the engine.

4.13.2 Install a driver

To install a driver in SpagoBI you must simply put the driver packaged in \EXO-HOME\spagobi\WEB-INF\lib folder. Pay attention that the driver installation is not enough to use it in SpagoBI. To enable the drivers it's necessary to associate it to an engine, using the SpagoBI Engine Service in the administration portlet.

4.13.3 The Jasper Report Engine Installation

SpagoBI provides a report engine that uses the open source jasper report library to produce reports and it consists in a simple web application that will be invoked by SpagoBI using the specific Jasper Report Driver. You can find both the driver (**SpagoBIJasperReportDriver.zip**) and the engine web application (**SpagoBIJasperReportEngine.zip**) in the SpagoBI distribution.
To start using the Jasper Report Engine do the following steps:

- install the driver on SpagoBI: extract from the **SpagoBIJasperReportDriver.zip** the jar file **sbi.drivers.jasperreport.jar** and put it into into the **SPAGOBI-HOME\WEB-INF\lib** folder if you're using the eXo tomcat express edition else copy in **EXO-EAR-HOME** directory.

- install the engine application: extract from the **SpagoBIJasperReportEngine.zip** the war file **SpagoBIJasperReportEngine.war** and deploy it in an application server. It's possible to install the web application in the same or in a different server in which SpagoBI is deployed, this is because it is a J2EE compliant web application.
- configure the engine into the SpagoBI engine administration service registering the complete name of the driver class and the engine application url.

Installing this Engine in Tomcat do not create particular problems because you must simple put the war file in **EXO-TOMCAT-HOME.** JBoss and Jonas requires further steps described in the next subsections.

Installing SpagoBIJasperReportEngine.war on Jboss 4.01

- Unzip the war to have a folder called **SpagoBIJasperReportEngine.war**
- Copy the folder on **JBOSS_HOME\server\default\deploy**
- Remove the following jar files from WEB-INF\lib

  commons-logging-api-1.0.2.jar

  commons-logging-1.0.2.jar

  xalan.jar

  xercesImpl.jar

  xmlParserAPIs.jar

  log4j-1.2.8.jar

- Prepare a file jboss-web.xml with the following content:

```
<jboss-web>
    <class-loading java2ClassLoadingCompliance="false">
    <loader-repository>
    dot.com:loader=SpagoBIJasperReportEngine
    <loader-repository-config>java2ParentDelegation=false</loader-repository-config>
    </loader-repository>
    </class-loading>
    <resource-ref>
    <res-ref-name>jdbc/sbifoodmart</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <jndi-name>java:/sbifoodmart</jndi-name>
    </resource-ref>
```

```
</jboss-web>
```

- Remove in the file web.xml the reference to the jn di resource for JSR170 repository because in jboss we don't use it JSR170 deployed as JNDI resource.

Installing SpagoBIJasperReportEngine.war on Jonas 4.3.5

- Unzip the war to have a folder called **SpagoBIJasperReportEngine.war**
- Copy the folder on **JONAS-HOME\webapps\autoload**
- Remove the following jar files from WEB-INF\lib

  commons-logging-api-1.0.2.jar

  commons-logging-1.0.2.jar

  xalan.jar

  xercesImpl.jar

  xmlParserAPIs.jar

- Prepare a file jonas-web.xml with the following content:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<jonas-web-app xmlns="http://www.objectweb.org/jonas/ns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.objectweb.org/jonas/ns
http://www.objectweb.org/jonas/ns/jonas-web-app_4_0.xsd" >
<jonas-resource>
<res-ref-name>jdbc/sbifoodmart</res-ref-name>
<jndi-name>jdbc/sbifoodmart</jndi-name>
</jonas-resource>
<java2-delegation-model>false</java2-delegation-model>
</jonas-web-app>
```

- Remove in the file web.xml the reference to the jn di resource for JSR170 repository because in jonas we don't use it JSR170 deployed as JNDI resource.

4.13.4 The Jpivot Engine Installation

SpagoBI provides an Olap Analysis and Pivoting engine that uses the open source jpivot product an it consists in a web application that will be invoked by SpagoBI using the specific Jpivot Engine Driver. You can find both the driver (**SpagoBIJPivotEngineDriver.zip**) and the engine web application (**SpagoBIJPivotEngine.zip**) in the SpagoBI distribution.
To start using the Jasper Report Engine do the following steps:

- install the driver on SpagoBI: extract from **the SpagoBIJPivotEngineDriver.zip** the jar file **sbi.drivers.jpivot.jar** and put it into the **SPAGOBI-HOME\WEB-INF\lib** folder if you're using the eXo tomcat express edition else copy in **EXO-EAR-HOME** directory.
- install the engine application: : extract from the **SpagoBIJPivoEngine.zip** the war file **SpagoBIJPivotEngine.war** and deploy it in an application server. It's possible to install it in the same or in a different server in which SpagoBI is deployed because it is a J2EE compliant web application.
- configure the engine into the SpagoBI engine administration service registering the complete name of the driver class and the engine application url.

Installing this Engine in Tomcat do not create particular problems because you must simple put the war file in **EXO-TOMCAT-HOME.** JBoss and Jonas requires further steps described in the next subsections.

Installing SpagoBIJPivotEngine.war on Jboss 4.01

- Unzip the war to have a folder called **SpagoBIJasperReportEngine.war**
- Copy the folder on **JBOSS_HOME\server\default\deploy**
- Remove the following jar files from WEB-INF\lib

  commons-logging.jar

  commons-logging-api.jar

  log4j-1.2.8.jar

- Prepare a file jboss-web.xml with the following content:

```
<jboss-web>
    <class-loading java2ClassLoadingCompliance="false">
    <loader-repository>
    dot.com:loader=SpagoBIJpivotEngine
    <loader-repository-config>java2ParentDelegation=false</loader-repository-config>
    </loader-repository>
    </class-loading>
```

```
<resource-ref>
<res-ref-name>jdbc/sbifoodmart</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<jndi-name>java:/sbifoodmart</jndi-name>
</resource-ref>
</jboss-web>
```

- Remove in the file web.xml the reference to the jn di resource for JSR170 repository because in jboss we don't use it JSR170 deployed as JNDI resource.

Installing SpagoBIJPivotEngine.war on Jonas 4.3.5

- Unzip the war to have a folder called **SpagoBIJasperReportEngine.war**
- Copy the folder on **JONAS_HOME\server\default\deploy**
- Remove the following jar files from WEB-INF\lib

  commons-logging.jar

  commons-logging-api.jar

- Prepare a file jonas-web.xml with the following content:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<jonas-web-app xmlns="http://www.objectweb.org/jonas/ns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.objectweb.org/jonas/ns
http://www.objectweb.org/jonas/ns/jonas-web-app_4_0.xsd" >
<jonas-resource>
<res-ref-name>jdbc/sbifoodmart</res-ref-name>
<jndi-name>jdbc/sbifoodmart</jndi-name>
</jonas-resource>
<java2-delegation-model>false</java2-delegation-model>
</jonas-web-app>
```

- Remove in the file web.xml the reference to the jn di resource for JSR170 repository because in JOnAS we don't use it JSR170 deployed as JNDI resource.

4.13.5 Configure connections for Jasper Report and Jpivot Engines

Both Jasper Report and Jpivot Engines need a connection to a database metadata for retrive the data to fill the report or the olap model. The connection to database for these two engines can be configured in two different xml file:

- **JasperReportApplication\WEB-INF\classes\engine-config.xml** for Jasper Report engine
- **JpivotApplication\WEB-INF\classes\connections-config.xml** for Jpivot Engine

These two files have the same sintax apart from the initial xml envelope which for the first is <ENGINE-CONFIGURATION> and for the second is <CONNECTIONS-CONFIGURATION>. Into the external envelop both file defines the connection in the same way which is described after.
Each request to an engine will use one of all the connections defined, if the request contains a "connectionName" parameter the engine try to use the connection with name equals to the parameter value, if the request as no "connectionName" parameter, or the name specified doesn't exist, the engine will use the default connection.
The connection configuration can be of two types, the first try to retrive the connection from a jndi connection pool, the second try to establish a direct connection to a database using the standard jdbc mechanism of driver and database url string.
The two different configuration are presented below:

```
<CONNECTION name="fill with the connection logical name"
            isDefault="true/false" isJNDI="true"
            initialContext="fill with the root jndi context
                            example java:comp/env"
            resourceName="fill with the jndi resource name"/>
<CONNECTION name="fill with the connection logical name"
            isDefault="true/false" isJNDI="false"
            driver="complete driver class name"
            user="database user name" password="database user password"
            jdbcUrl="database connection string"/>
```

5 Using SpagoBI with eXo Portal Server

SpagoBI consists of a set of portlets compliant to the jsr 168 specificantion and so to stat use it it's necessary to import the portlets into a portal server. This document talk about the installation in a eXo tomcat portal server and so in the rest of this section will be describe an example of SpagoBI configuration for it. Remember that it's possible to configure pages and portlets of a portal server in a infinite number of different combinations so this explaniation want only to be an example to introduce you to the SpagoBI portlets.

5.1 Import SpagoBI portlets

To import spagobi portlets inside exo platform open exo and connect with the administrator account, which is usually admin/exo (see Figure 1). Once successfully connected, click the portlet registry (***Portal/Portlet Registry***) in navigation bar on the left. (see Figure 2)
In the portlet registry user interface it's possible to import portlets and to do this simply click on the Import Portlets button. If all is ok you can see the spagobi portlets on the left where all available portlets in the portal are shown.

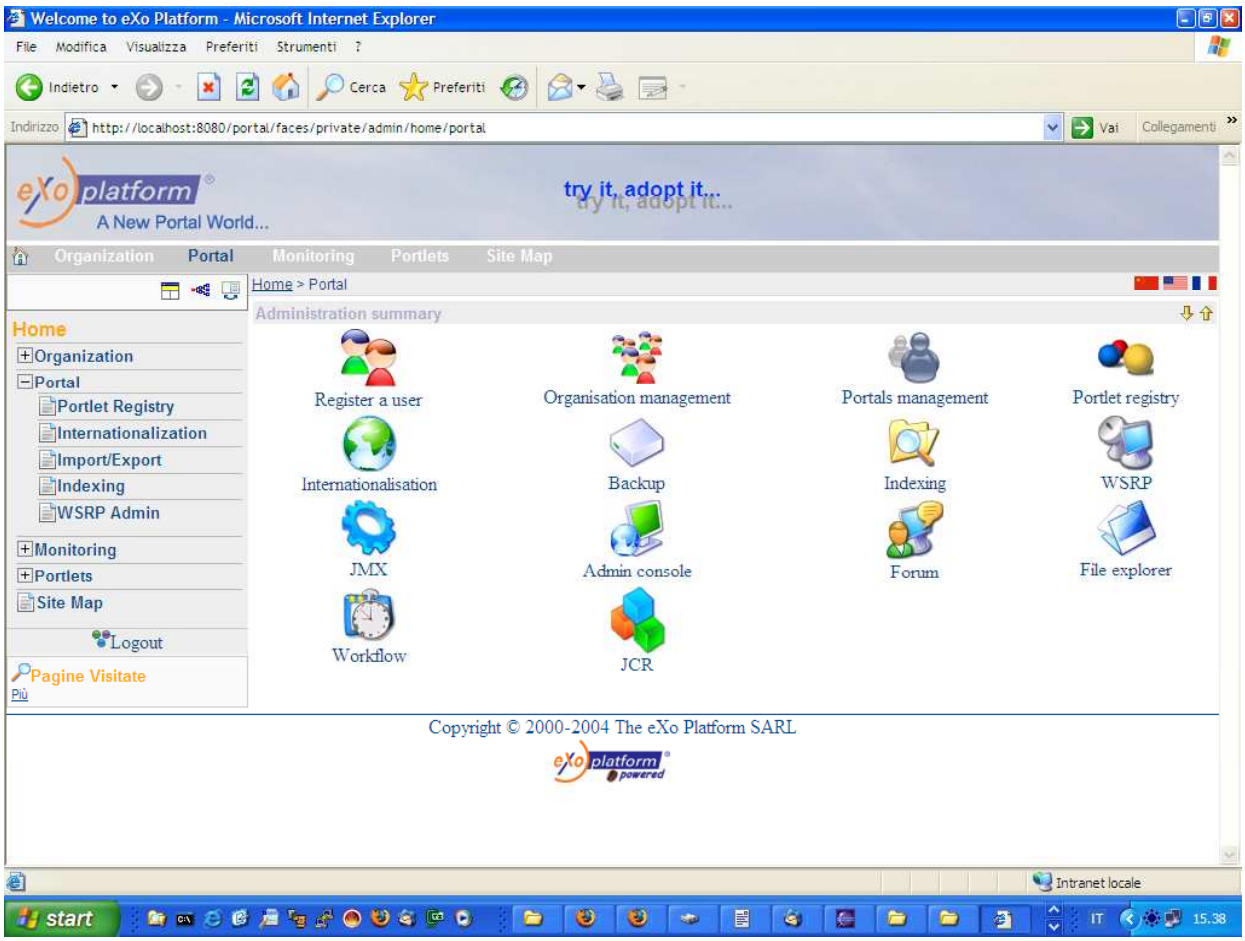


Figura 1

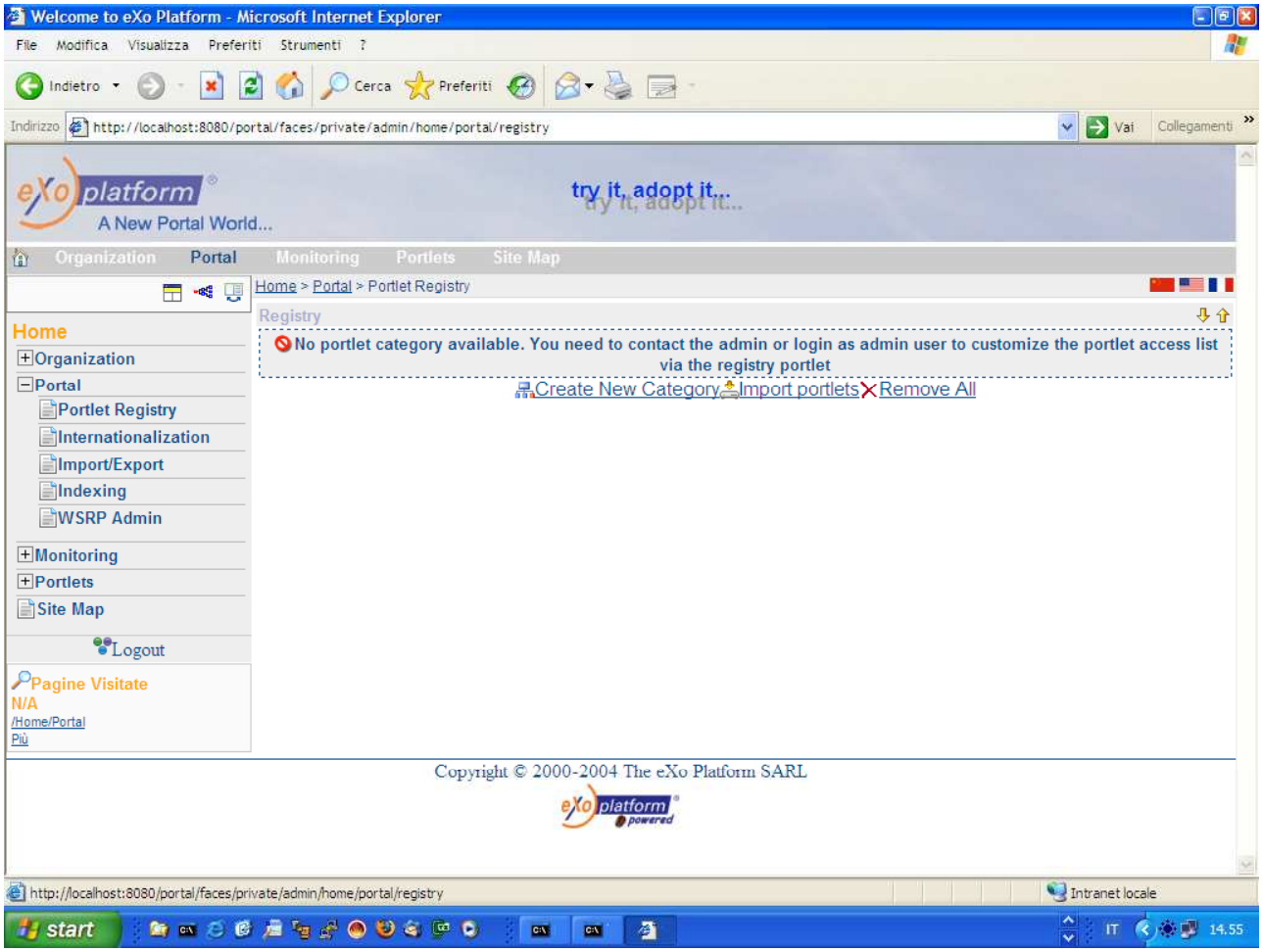


Figura 2

5.2 Configure Roles and Accounts

SpagoBI is an application designed to be used by users that have different roles in organizations. In particular spagobi will be used by:

- business documents developers.
- business testers.
- final users (usually company managers and final users of reports and olap analysis)

in a simple configuration you can provide tree portal roles, one for each category previous defined. To configure roles in eXo Portal connect to eXo with administrator account and go to Portal from left navigation bar (see Figure 1). Then click on ***Organisation management*** (see Figure 3).

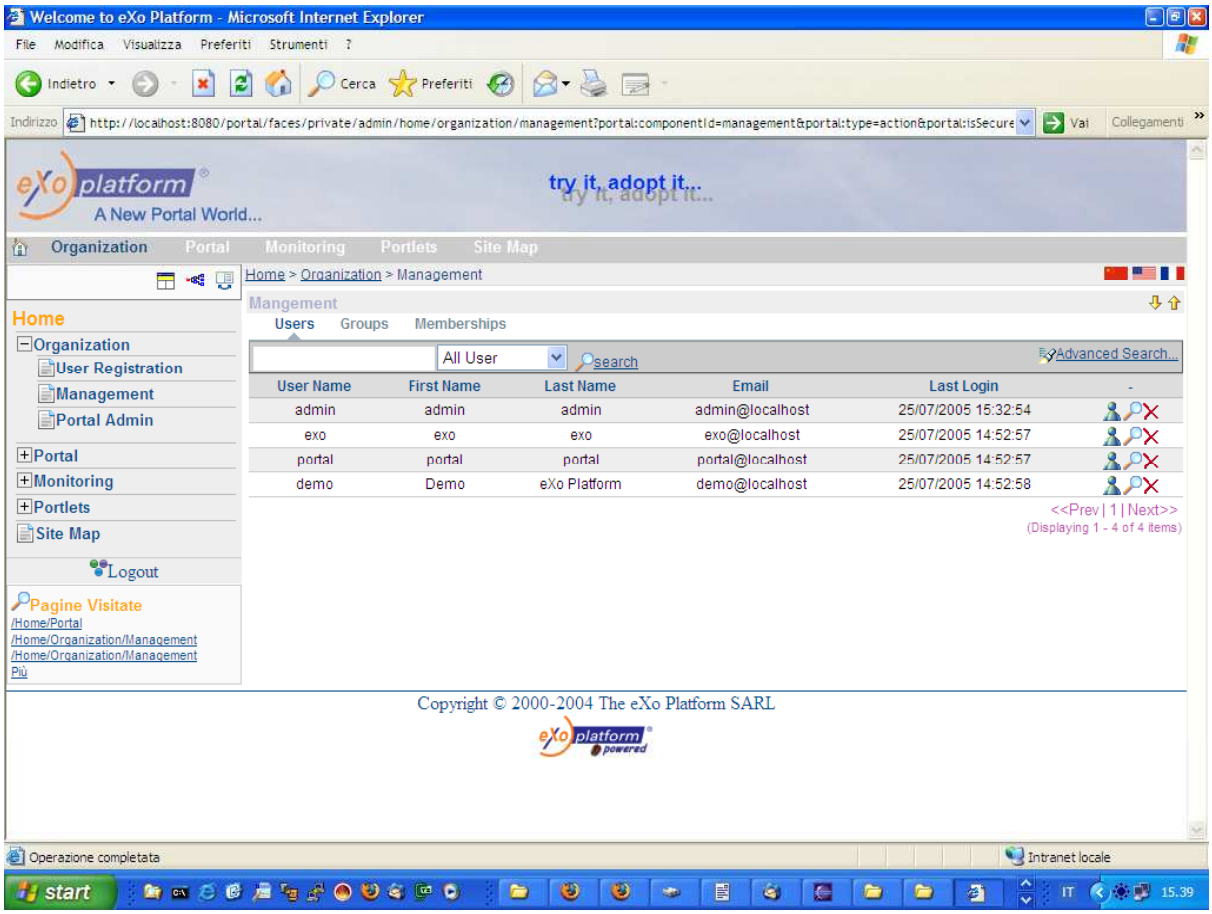


Figura 3

To define a new role click on ***Group*** an then on ***Add Group*** (see Figure 4) and insert three groups:

- biusers
- bitesters
- bidevs

Once finished with the groups (roles) define at least three users. Always in the ***Organisation Management*** go on the users tab (see Figure 5) and define the following users (remember passwords)

- dev/dev
- user/user
- test/test

After the user definitions make the following associations from ***Organisation Management Tab*** (see Figure 6)**.**

- Assign dev user to biusers group
- Assign test user to bitesters group
- Assign user user to biusers group

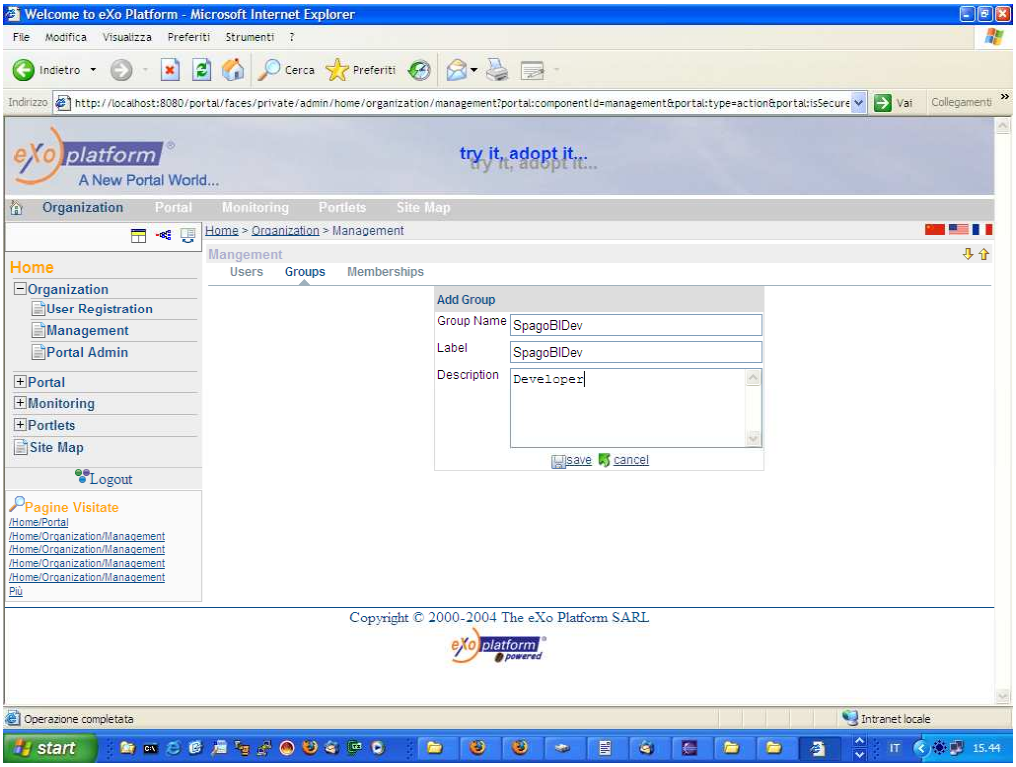


Figura 4

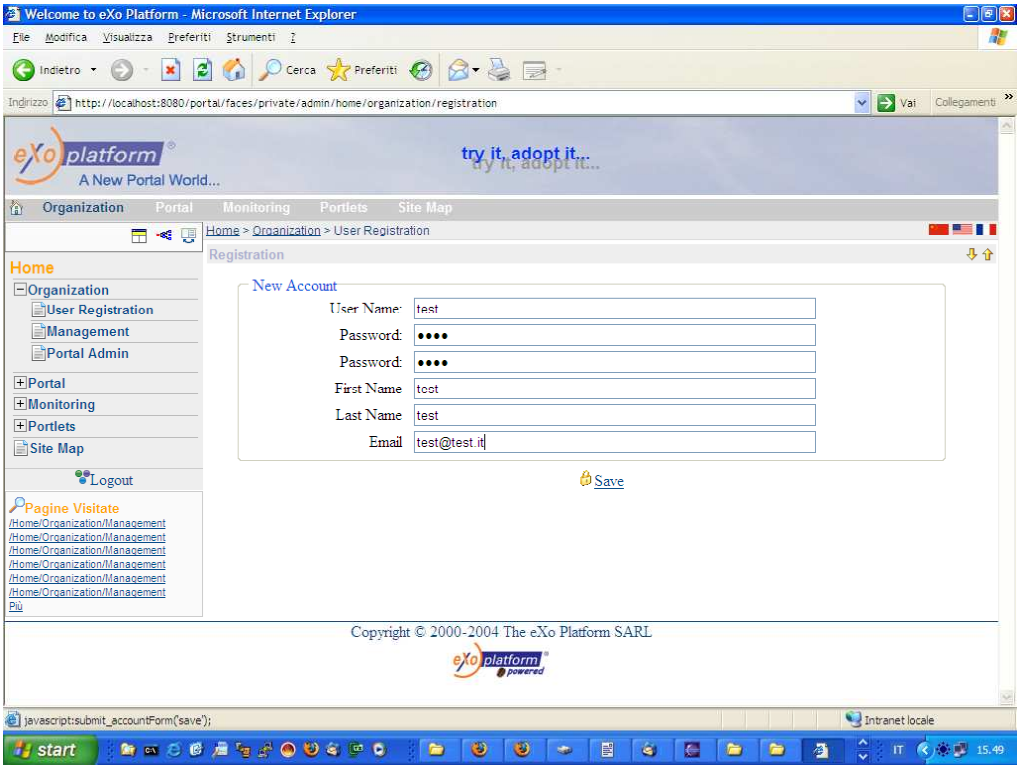


Figura 5

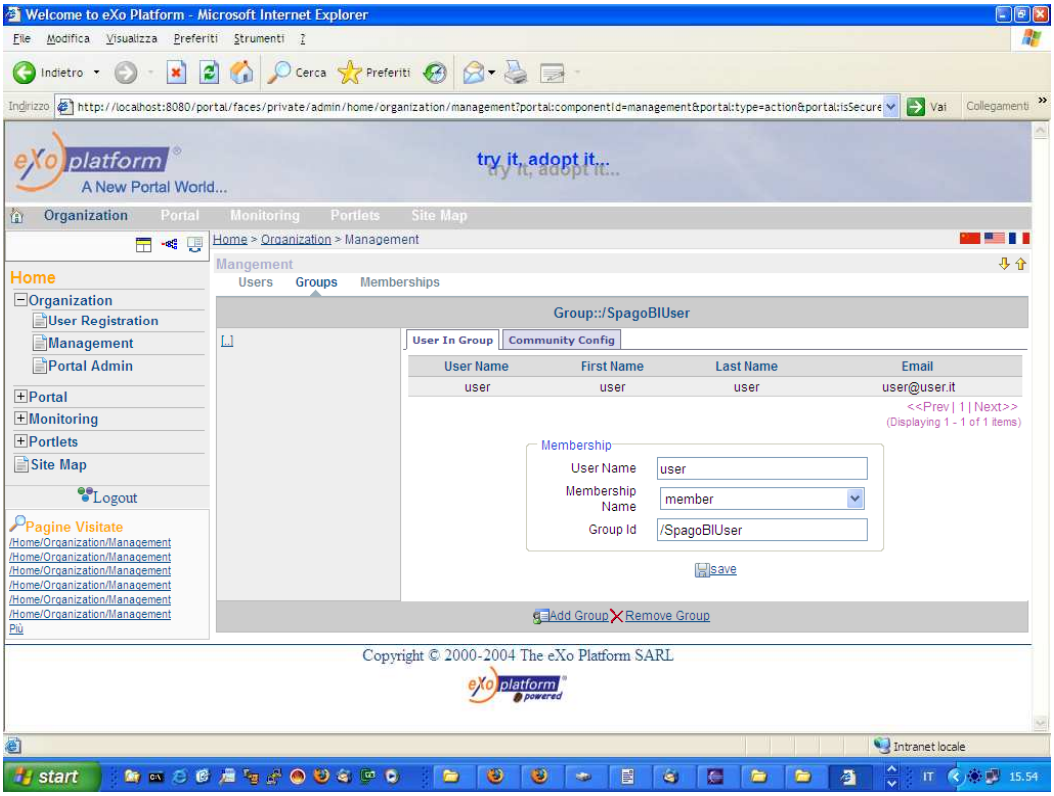


Figura 6

5.3 Configure Portal Pages for each user

After the user and group/role configuration proceed to customize the portal in order to define the home page for each users that logs into SpagoBI. The home page must contains the correct SpagoBI portlet based on the four general roles, admin, developer, tester and user.

- The developer user (dev) must see the SpagoBI development portlet.
- The tester and the final user must see the SpagoBI Functionality portlet. (Note that is the spagobi admin, in our configuration admin/exo, that map a portal role to a tester or to a final user when the functionality tree is created and configured)
- The admin user must see the SpagoBI Admin Portlet.

To configure the portal pages for each users, log yourself into exo portal server as admin/exo (see Figure 1), go to ***Portal*** and click on ***Portals Management Button*** (see Figure 7). Here you can edit and change the portal configuration for each user, so:

- For admin add everywhere you want the SpagoBISettings Portlet.
- For Testers and Users add everywhere you want the SpagoBIFunctionality portlet.
- For developers add everywhere you want the SpagoBIDevelopment Portlet.

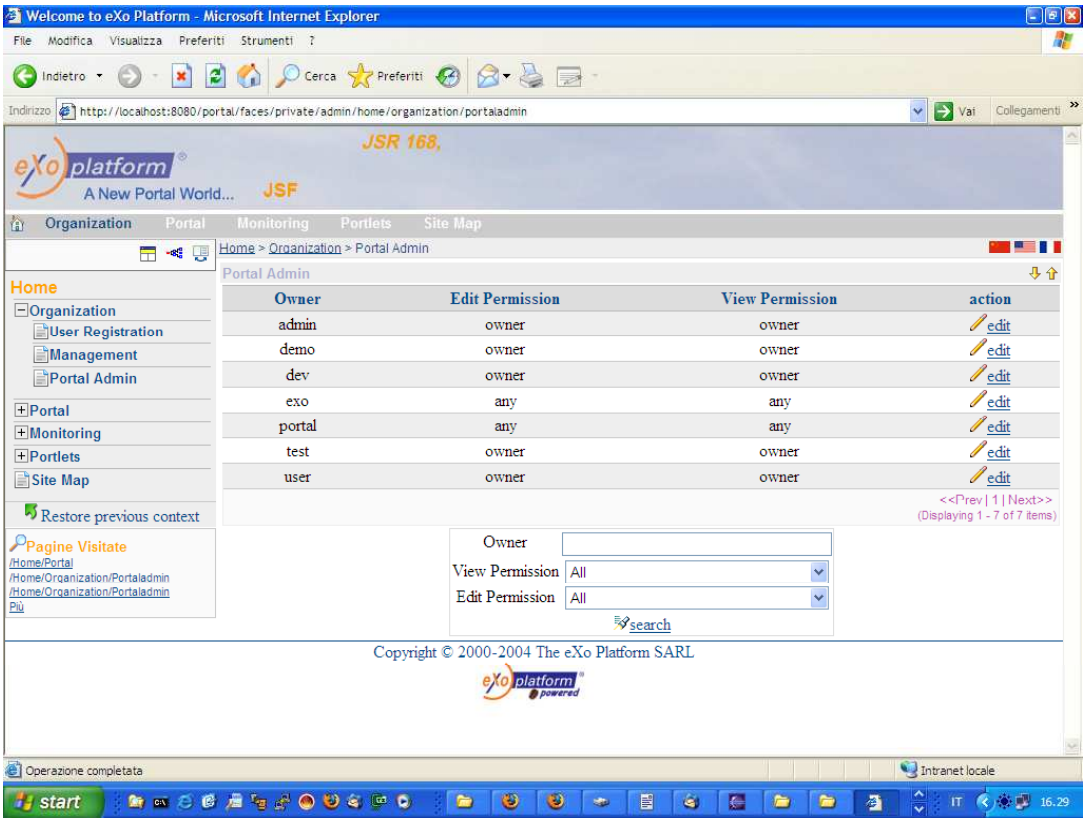


Figura 7

5.3.1 Configure Portal for administrators

Connect as admin/exo and from Portal/Organisation Management (see Figure 7) click on edit button for admin user. Now it's possible to change the portal of the admin (refer to eXo documentation) and add the SBISetting Portlet where you prefer. If you do all correctly when someone log in as the admin user the SBISetting portlets must be visible. (see Figure 8)

5.3.2 Configure Portal for developers

Connect as admin/exo and from Portal/Organisation Management (see Figure 7) click on edit button for dev user. Now it's possible to change the portal of the developer role (refer to eXo documentation) and add the SBIDevelopmentContext Portlet where you prefer. If you do all correctly when someone log in as the dev user the SBIDevelopmentContext portlet must be visible. (see Figure 9).

5.3.3 Configure Portal for final users and testers

Connect as admin/exo and from Portal/Organisation Management (see Figure 7) click on edit button for test and final user. Now it's possible to change the portal for final users and testers (refer to eXo documentation) and add the SBIFunctionality Portlet where you prefer. If you do all correctly when someone log in as the tester or final user the SBIFunctionality Portlet portlet must be visible.

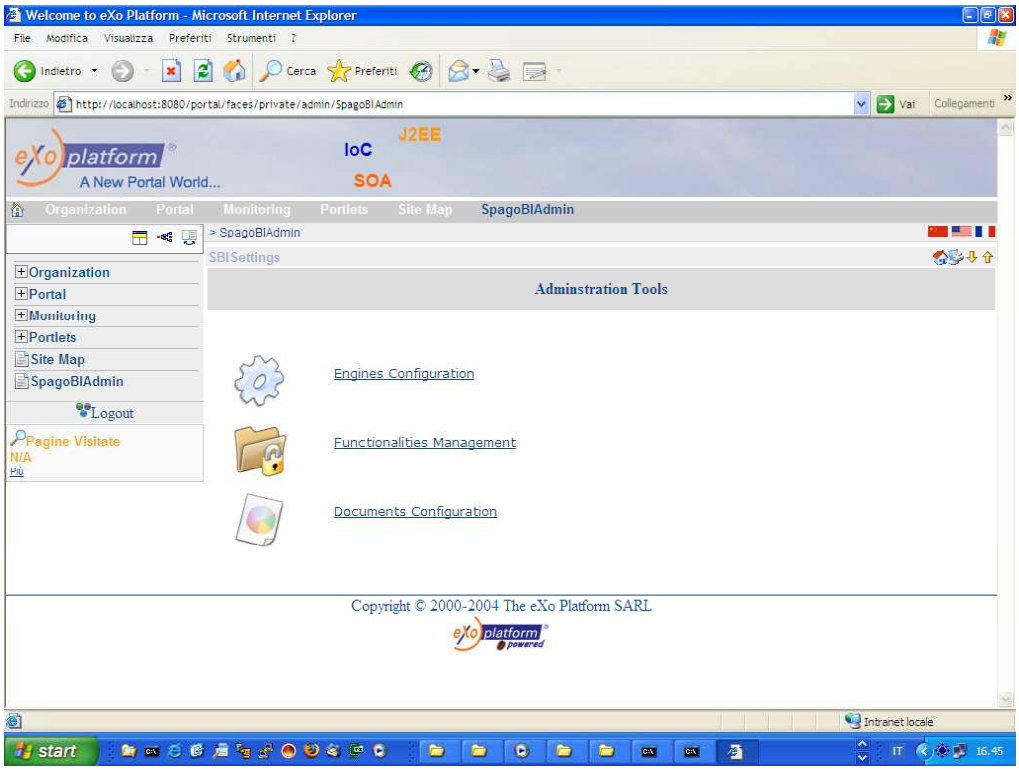


Figura 8

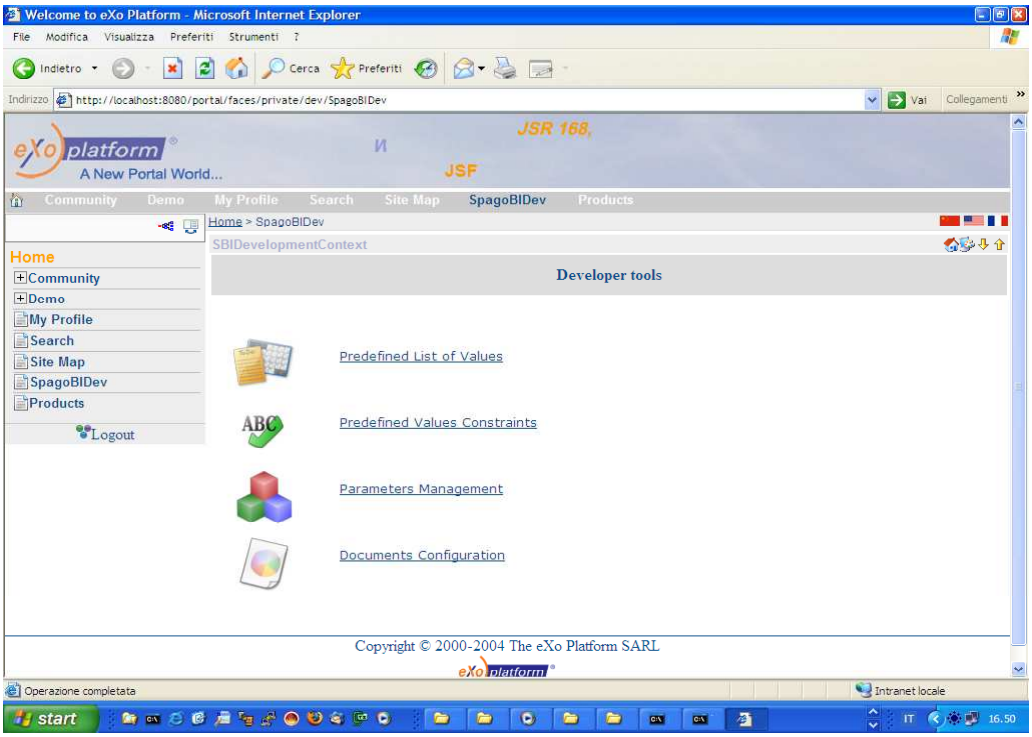


Figura 9

6 Common Problems

6.1 Add new database connection to SpagoBI

In order to add a new database connection to SpagoBI you need to change the file data_access.xml in EXO_HOME/webapps/spagobi/WEB-INF/conf directory. This file contains all the configurations for every database accessible to SpagoBI. It's possible to retrive the datasource from a jndi tree or to create a new one defining the implementation class and his parameter.
For the jndi configuration you need to modify the EXO-HOME/conf/server.xml file in order to add a new global name datasource (see “Configuring JNDI Resources” section), add into the EXO-TOMCAT/conf/locahost/Catalina/spagobi.xml file the link to the global datasource, and after, in the data_access.xml, put the following xml envelope:

```
<CONNECTION-POOL connectionPoolName="spagobi"
                 connectionPoolFactoryClass=
                 "it.eng.spago.dbaccess.pool.JNDIConnectionPoolFactory"
                 connectionPoolType="native">
      <CONNECTION-POOL-PARAMETER
            parameterName="initialContext"
            parameterValue="java:comp/env"/>
      <CONNECTION-POOL-PARAMETER
            parameterName="resourceName"
            parameterValue="name of the jndi resource"/>
      <CONNECTION-POOL-PARAMETER
            parameterName="driverVersion"
            parameterValue="2.1"
            parameterType=""/>
      <CONNECTION-POOL-PARAMETER
            parameterName="sqlMapperClass"
            parameterValue=
            "it.eng.spago.dbaccess.sql.mappers.OracleSQLMapper"
            parameterType=""/>
</CONNECTION-POOL>
```

If you wanto to configure a new datasource without defining it like a jndi resource you need to put into the data_access.xml the following xml envelope:

```
<CONNECTION-POOL connectionPoolName="spagobi"
                 connectionPoolFactoryClass="it.eng.spago.dbaccess.pool.DBCPCon
                 nectionPoolFactory" connectionPoolType="native">
      <CONNECTION-POOL-PARAMETER parameterName="connectionString"
            parameterValue="connection string for the database"
            parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="driverClass"
            parameterValue="complete driver class name" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="driverVersion"
            parameterValue="2.1" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="user"
```

```
            parameterValue="user name" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="userPassword"
            parameterValue="password" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="poolMinLimit"
            parameterValue="1" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="poolMaxLimit"
            parameterValue="10" parameterType=""/>
      <CONNECTION-POOL-PARAMETER parameterName="cacheTimeToLiveTimeout"
            parameterValue="10" parameterType="" />
      <CONNECTION-POOL-PARAMETER parameterName="cacheInactivityTimeout"
            parameterValue="10" parameterType="" />
      <CONNECTION-POOL-PARAMETER parameterName="sqlMapperClass"
            parameterValue="it.eng.spago.dbaccess.sql.mappers.OracleSQLMapper"
            parameterType=""/>
</CONNECTION-POOL>
```

Each datasource defined must also be registered into the Connection Manager otherwise SpagoBI isn't able to connect to it, so remember for each one of them to put into the `<CONNECTION-MANAGER>` tag of the data_access.xml one tag with his name.

```
<CONNECTION-MANAGER>
      <REGISTER-POOL registeredPoolName="name pool 1"/>
      ....
      <REGISTER-POOL registeredPoolName=”name pool n"/>
</CONNECTION-MANAGER>
```

All the datasources configured and registered will be accessible to SpagoBI and it's possible to use them like connection name for the query lov. If you need to define a datasource in a different manner you can look at the Spago framework documentation for data_access configuration.

6.2 Add a new language

The static interface of spagoBI supports multi-language and the default languages are english and italian. If you need to add a new language you must:

- add the new language to the exo portal (see eXo platform documentation).
- edit the EXO-TOMCAT/webapps/spagobi/WEB-INF/classes/messages_en_US.properties file, traduce the label values contained into your language and save it in the same folder renaming it in the following manner: messages_{languageCode}_{countryCode}.properties.
- Edit the file EXO-TOMCAT/webapps/spagobi/WEB-INF/conf/spagobi/spagobi.xml and add to the tag <LANGUAGE_SUPPORTED> an xml envelope like this: <LANGUAGE default="false" language="**language_code**" country="**country_code**" />

6.3 Change Security Keys

The SpagoBIJasperReportEngine and SpagoBIJPivotEngine engines and their relative drivers implements an authentication mechanism using a Public/Private Key pair (1024 bits for DSA algorithm). The Private Key is owned by SpagoBI, which use it to sign the requests, while the

Public Key is owned by the engines, which use it to verify the firm of the requests. SpagoBI Distribution contains a default Key pair but it's possible to change it.
In order to generate a new key pair you have to:

- download the SpagoBIComponents binary distribution;
- unzip the package (into the SpagoBIKeyGenerator subfolder you will find a SpagoBIKeyGenerator-1.0.jar file);
- from the SpagoBIKeyGenerator subfolder, run the command 'java –cp SpagoBIKeysGenerator-1.0.jar it.eng.spagobi.security.KeyGenerator'; the tools interface must be showed (Figure 10);
- fill the field 'Key Prefix' with the prefix for the two file of the key pair. The name of the two files generated will be 'prefix_privatekey' and 'prefix_publickey';
- fill the field 'Save Folder' with the folder path where the key pair files will be created;
- Press the button 'Generate Key'. The interface will disappear and the keys file will be generated.

Once Generated the keys it's necessary to set them into SpagoBI, as follows:

- copy the private key file into the directory <*spagobi_webapps_dir*>/WEB-INF/classes;
- edit the file <*spagobi_webapps_dir*>/WEB-INF/conf/spagobi/spagobi.xml and insert the name of the private key file into the <SPAGOBI_PRIVATE_KEY_DSA /> tag;
- copy the public key file into the directories :
  - <*spagobi_JasperReportEngine_webapps_dir*> /WEB-INF/classes
  - <*spagobi_JpivotEngine_webapps_dir*>/ WEB-INF/classes
- edit the file <*spagobi_JasperReportEngine_webapps_dir*>/WEB-INF/classes/security-config.xml and insert the name of the public key file into the <SPAGOBI_PUBLIC_KEY_DSA/> tag;
- edit the file <*spagobi_JpivotEngine_webapps_dir*>/WEB-INF/classes/security-config.xml and insert the name of the public key file into the <SPAGOBI_PUBLIC_KEY_DSA /> tag.

After the reload of the engines and spagobi application the system will use the new key pair.

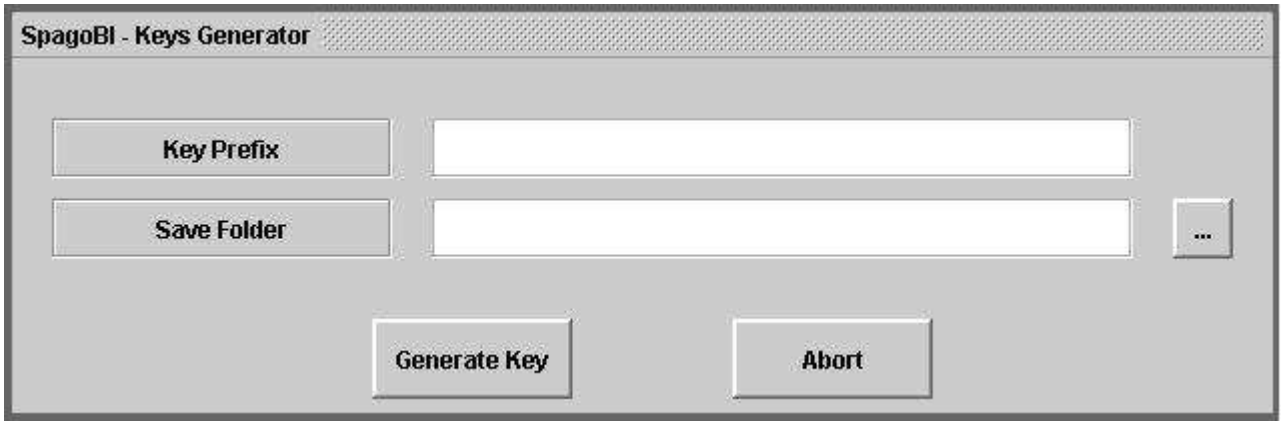


Figura 10