Kuwaiba Open Inventory Administrator's Manual

Neotropic SAS

27.07.2016

System Version **1.0**
Visit **kuwaiba.org** for documentation, latest updates and upcoming events

Contents

Document History

Date	Comments
January 19th 2011	First version. It uses the number 0.3 just to keep compatibility with the version used for the user manual
March 17th 2011	Corrections regarding to the changes made in the version 0.3 beta 2
November 8th 2011	Added details related to the database setup. Added a troubleshooting section
May 22nd 2012	Adaptation to version 0.4
July 13th 2012	Multiple fixes and clarifications
December 12th 2013	Style corrections
January 19th 2015	Corrected how the rmiregistry is launched
July 27th 2016	Adapted to Kuwaiba version 1.0. LaTeX is now used instead of LibreOffice to create the documentation.

License

Disclaimer

Introduction

Kuwaiba is a plain client/server application. It's Java on both sides, though it's fairly easy to implement clients in any language, since the server exposes a comprehensive SOAP web service interface, which is the native way it talks to the default client and third-party applications.
In the figure below, you can see a simplified block diagram of the server application.

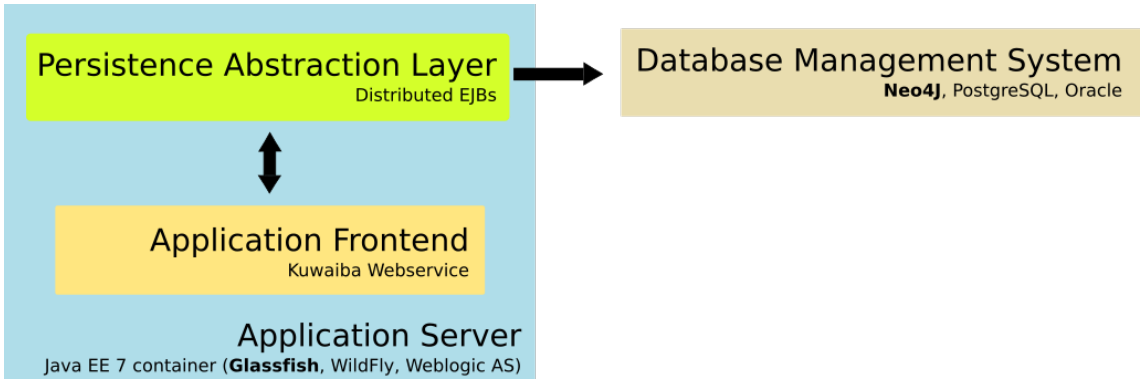

Figure 1: Simplified server architecture

We won't go into technical details here, suffice to say that there are three main blocks that are important to keep in mind, as all the troubleshooting procedures will revolve around them. The lower level component is the database. Kuwaiba uses a graph-oriented database called Neo4J. After a few releases using a conventional relational database, we realized that the best way to model a telecommunications network is not a set of related tables, but a graph. Neo4J may work in two modes: as a server responding JSON requests on a TCP port or as an embedded database, when the application access the database files directly without any service mediating between the application and the data. Kuwaiba uses the latter. This means that you can't access the database while Kuwaiba is running, because it will be locked. Running Neo4J in embedded mode increases the performance, sacrificing accessibility to the data.

The middle layer is called the Persistence Service. This is one of the most important components, and it's the responsible of accessing the information stored in the database in a safe way, isolating that logic from the res of the application. Other layers of the application will never know that the backend is Neo4J thanks to this module, and this enables the development of implementations using other DBMS like Oracle, for example. It's called a service for historic reasons: Prior to version 1.0 this component was a standalone application that ran in a different JVM than the one used by the application server. This was complicated to setup, so it was merged with the application running on the application server. However, It still runs when the application is deployed or at application server (AS) startup. You should check the AS log files to see if the Persistence Service is running correctly, or nothing else will work.

The third component is the frontend of the server. It implements the web service that allows the communication between the server and its clients or third party applications. In a general sense, it's a northbound interface. As we will see later, you will be able to access a simple web-based interface to perform administrative tasks, like resetting passwords or creating the default database structure. Also, you will be able to access the WSDL that would eventually allow you to integrate in-house applications.

As you can see, this is pretty straightforward and once it's done the first time, it can be easily reproduced. Starting from scratch, given that all requirements are already downloaded, the server can be configured and started in less than 10 minutes.

Server Installation

The server application can be installed on any platform supported by Java EE in all its implementations (OpenJDK, Oracle's JDK, etc). The instructions are basically the same for all platforms, and the differences will be highlighted when necessary.

Requirements

- JDK 7 or superior. On Windows platforms, you'll need to download it from the Oracle's website[1], while on Linux or BSDs systems you can use the package/port managers to install OpenJDK from a repository. Refer to the documentation of your distribution for more details on how to install Java.

> **Important**
> Note that you need the JDK (Java Development Kit), not only the JRE (Java Runtime Environment).

- An application server. The instructions presented in this document apply to Glassfish version 3 or above, however, Glassfish 4.1 is highly recommended. You can find details on its installation process here[2]. In short, just download the .exe/.bin installer from its official site[3] and execute it choosing the default options. Make sure you secure and tune it in your production environment. You can also use any other Java EE 7 compliant application server, but this document will use Glassfish.
- Neo4J 2.3.x Community Edition. We just need some libraries included in its installation package, but it's good to have it all installed so we can use it for troubleshooting problems in the future. You can get the installer from here[4].

> **Important**
> Neo4J libraries are no longer distributed in the server installation package due to open source licenses conflicts.

- The latest Kuwaiba Server installation package. You can get it from here[5]. As of version 1.0, both client and server are in the same file, whose name must look like this

  `kuwaiba\_[version]\_[alpha/beta/stable].zip`

[1] Oracle's JDK Downloads http://www.oracle.com/technetwork/java/javase/downloads/index.html
[2] Glassfish Quick Start Guide https://glassfish.java.net/docs/4.0/quick-start-guide.pdf
[3] Glassfish Official Web Site https://glassfish.java.net/
[4] Neo4J 2.3.2 Community Edition Downloads https://neo4j.com/download-thanks/?edition=community&flavour=unix&release=2.3.2
[5] Kuwaiba Installation Packages http://sourceforge.net/projects/kuwaiba/files/

Step by Step Guide

Step 1: Preparation

Let's take a look at the structure of the server installation package:

dbs	In this directory you will find default databases you can use to get started with Kuwaiba. • **01_empty_kuwaiba.db.zip**: Contains the database schema plus a default user account. • **02_containment_kuwaiba.db.zip**: Contains the database schema, a default user account and a sample containment hierarchy but no objects. • **02_data_kuwaiba.db.zip**: Contains the same as 02_containment_kuwaiba.db plus some objects, views and tasks
KuwaibaServer.ear	The enterprise application to be deployed on the application server
README	A simple README file with useful information about where to get resources.
THIRDPARTY	The list of open source, third-party components used in the server, as well of their respective licenses and links to the original projects.
LICENSE.EPL	Text of the EPLv1 license
LICENSE.GPLv2	Text of the GPLv2 license
CHANGELOG	List of changes since the first version
class_hierarchy.xml	A default database schema that can be uploaded instead of using one of the default databases. Only for experimented users.

Before proceeding with the formal installation we must configure some things:

- Create the directory where the database will be stored. Make sure the user that will run Glassfish has read and write permissions on that directory. If you want to create the database from scratch, leave this directory empty, otherwise, unzip one of the default databases and copy it there.
- Create a directory to store the backgrounds and other files. Also make sure the user that will run Glassfish has read and write permissions on that directory.
- Open the file **KuwaibaServer.ear** and search for the file **KuwaibaServer-ejb.jar/META-INF/ejb-jar.xml**. Note that both **KuwaibaServer.ear** and **KuwaibaServer-ejb.jar** are actually .zip files with other extensions, so you should be able to open them with any zip manipulation utility. Modern programs allow you to modify the files contained within zip files and they will update the original compressed file.
  ejb-jar.xml is an XML file that contains configuration parameters:

Parameter	Description
dbUsername	Not used
dbPassword	Not used
dbPath	The path where the database is located. **Default value:** /data/db
dbHost	Not used
dbPort	Not used
backgroundsPath	The path where the backgrounds and other files are stored. **Default value:** /data/img/backgrounds
corporateLogo	The URL of the logo that will be displayed in the reports. **Default value:** http://neotropic.co/img/logo_small.png
enableSecurityManager	Enables the JVM security manager. This could be used to restrict the kind of code that can be executed. See **Appendix A. Security Manager Configuration** for details. **Default value:** false

Set the paths to appropriate values and compress the .ear and jar files if your archiving utility doesn't do it automatically. **Note:** You can also leave the default values, skip this step, and then modify the file once the whole bundle has been deployed on the application server. It will most likely be in the path:

```
[GLASSFISH_DIR]/glassfish/domains/domain1/[INSTALLER_FILE_NAME]
```

> **Important**
> If the path to the database does not exist or can not be written, the **Persistence Service won't start**. Check the **Troubleshooting** section for more details if this happens.

- Finally, copy all .jar files in the directory **lib** of the Neo4J's install package to the path

```
[GLASSFISH_DIR]/glassfish/domains/domain1/lib
```

Step 2: Deploying the application

- Open a command prompt/console and start Glassfish by typing this:

```
[GLASSFISH_DIR]/bin/asadmin start-domain domain1
```

- Once the application server is up and running, it will provide a web management interface listening on port 4848 (http://[server_domain_name_or_ip]:4848). Don't forget the credentials you provided during the server installation process because they will be used for further administration procedures. See figure 2.
- Now you are ready to deploy the enterprise application. Take the EAR file you will find in the server bundle called **KuwaibaServer.ear** and upload/publish it using the section Applications/Deploy. See figure 3.
- Test if the application was correctly deployed by opening the URL **http://[server_domain_name_or_ip]:8080/kuwaiba/**. See figure 4.

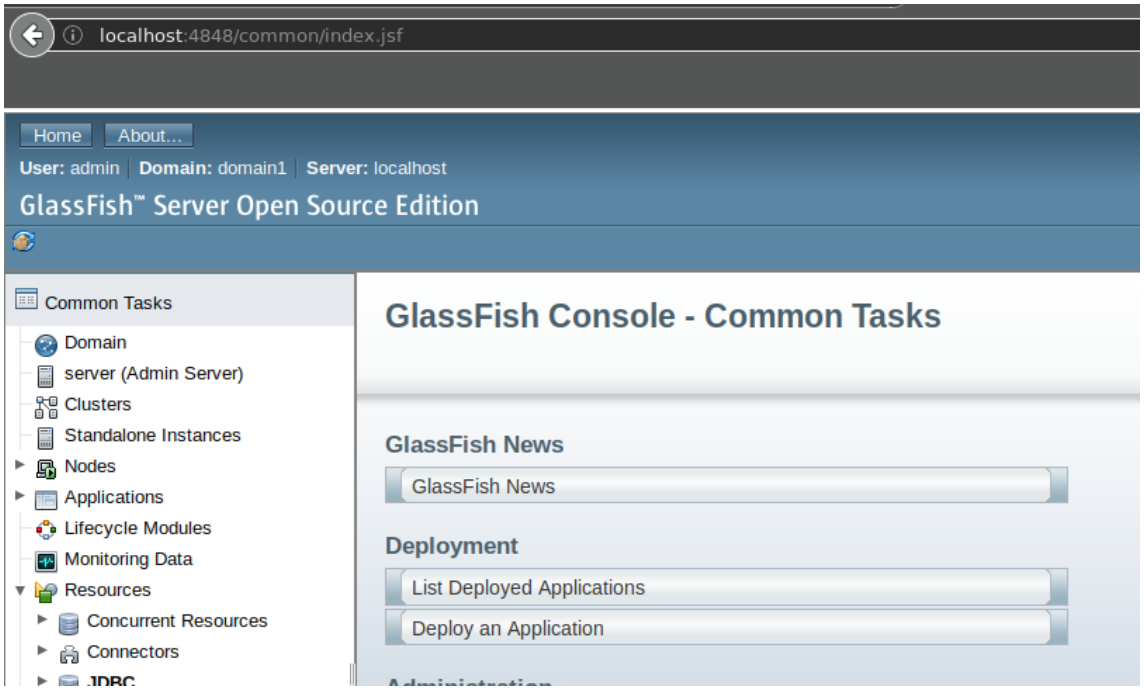

Figure 2: Glassfish management console

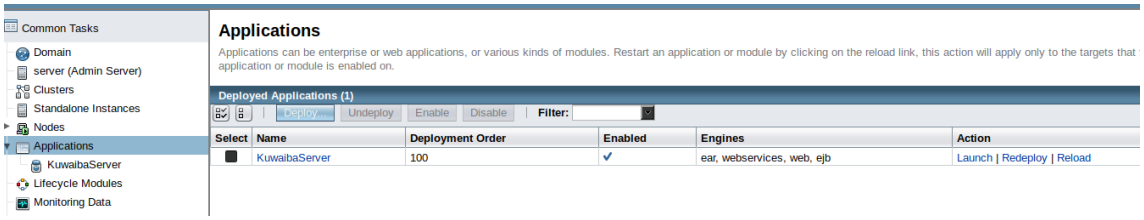

Figure 3: Deployed applications

> **Important**
> For security reasons, the administration console can **only** be opened from the same computer the server is running. If you try to open it from some other computer, you'll get a error.

- Additionally, **always** check the command prompt/console you used to launch the server to see if an error was thrown. Sometimes, the errors are logged only in the server log file, located at **[GF_INSTALL_DIR]/glassfish/domains/domainXX/logs/**. The application may have been deployed, but the Persistence Service might have not started.

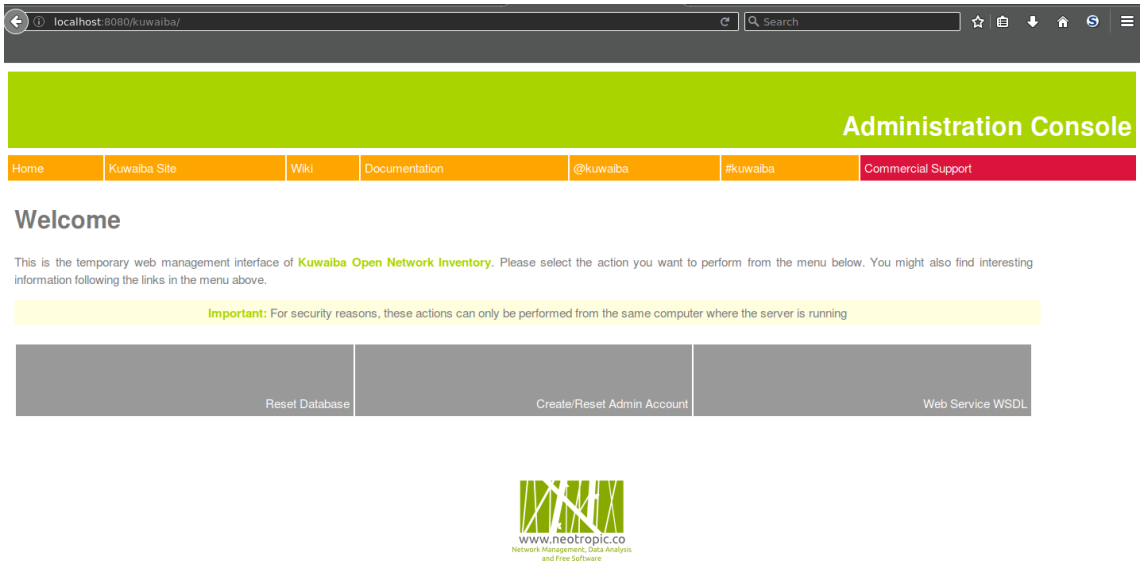

Figure 4: Administration console

Troubleshooting

1. If you get a "Cant't reach backend" error, it means that the Persistence Service didn't start. The most probable cause is that the database path was not set correctly, or the service does not have write access to it. Check Glassfish log files and search for the "[KUWAIBA]" tag to see the service startup messages.

2. A message "Can not find the class InventoryObject" means that

3. If you install Glassfish under MS Windows 7/Vista in **c:/glassfishv3**, check you have proper permissions to write on this location, since due to system restrictions it's not possible by default. That will affect the domain creation and application deployment processes.

4. In some cases, if you install Glassfish before to install the JDK under MS Windows (especially versions 7 and Vista) or when the installer detects a wrong version of Java, the variable AS_JAVA in **[GLASSFISH_INSTALL_DIR]/glassfish/config/asenv.bat** is set to a wrong location. It should be pointing to the JAVA_HOME value (this is, the JDK installation directory).

Client Installation

The client should work on all JRE supported platforms.

Requirements

- JRE 7 or superior

Step by step guide

1. Download the last stable client bundle. As explained in the past chapter, as of version 1.0, the client and server are included in the same .zip file.

2. Extract the client files.

3. Execute the binary suitable for your platform located at The file is called **kuwaibain-**

   `[EXTRACTION_DIR]/kuwaibainventory/bin`

   ventory.exe for Windows platforms and **kuwaibainventory** for Unix-like systems. [**EXTRACTION_DIR**] is the directory where you extracted the files.

4. You may need to change the file permissions to make the binary executable under Unix-like OS using the command

   ```
   chmod 755 [EXTRACTION_DIR]/kuwaibainventory/bin/kuwaibainventory
   ```

Appendices

Appendix A. Security Manager Configuration

If you have set the **enableSecurityManager** configuration variable to **true**,create a text file called **.java.policy** (note the dot before the name). Place it in your user directory (/home/[username] in UNIX environments, or **c:/Documents and Settings/[username]** or **c:/users/[username]** in Windows systems). It should contain these lines:

```
grant {
permission java.util.PropertyPermission "*","read,write";
permission java.util.PropertyPermission "sun.arch.data.model","read";
permission java.io.FilePermission "persistence.properties","read";
permission java.lang.RuntimePermission "accessDeclaredMembers", "read";
permission java.lang.RuntimePermission "shutdownHooks", "write";
permission java.lang.RuntimePermission "modifyThread";
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
permission java.net.SocketPermission "127.0.0.1","connect,accept,resolve";
permission java.io.FilePermission "<<ALL FILES>>","read,write,delete";
};
```

Note that you should NOT run Glassfish as root. These settings can be used if the persistence service AND the application run in the same box (the last line governs who can get connected to the service, in this case, only those applications running in the same computer). You might to add some more permissions depending on what other Java applications are you running on that computer. Also, the last two lines can be tuned to be less permissive, if preferred. For more information about permissions, please check the official documentation by Oracle[6] [7].
There's also some information on how to apply the security manager rules only to Glassfish here[8]

[6]Policy files http://docs.oracle.com/javase/1.3/docs/guide/security/PolicyFiles.html
[7]Permissions http://docs.oracle.com/javase/6/docs/technotes/guides/security/permissions.html
[8]The server.policy file https://docs.oracle.com/cd/E18930_01/html/821-2418/beabx.html

Appendix B. How to Configure a Secure Connection