

JORAM 3.7

INSTALLATION GUIDE

*Author: Frédéric Maistre.
November 6, 2003*

TABLE

1.	Introduction	3
2.	Getting JORAM	4
2.1.	<i>Installing a package</i>	4
2.2.	<i>Getting JORAM from CVS</i>	4
2.3.	<i>Directory structure</i>	4
2.4.	<i>Directories description</i>	6
3.	Compiling	8
3.1.	<i>Compiling the CVS tree</i>	8
3.2.	<i>Compiling the sources package</i>	9
3.3.	<i>Compiling JORAM samples</i>	10
3.4.	<i>Compiling kJORAM samples</i>	10
4.	Shipping JORAM	11
4.1.	<i>JORAM server</i>	11
4.2.	<i>Graphical tool</i>	11
4.3.	<i>JORAM client</i>	12
4.4.	<i>kJORAM client</i>	12
5.	Configuring Tomcat	13

1. Introduction

JORAM 3.7 includes:

- A **messaging server** (or MOM), providing the messaging functionalities: basically hosting and routing the messages exchanged by the client applications.
- A **JNDI** compliant naming server, persistent and reliable.
- **Client classes** allowing applications to access the MOM functionalities. Those interfaces are defined by the **JMS 1.1** specifications.
- A **graphical administration and monitoring tool**, allowing to modify and visualise the state of a JORAM platform (made of one or many interconnected servers).
- A specific set of classes usable by client applications running on a **J2ME** environment. This set of classes is JMS 1.1 like, and is called **kJORAM**.
- **Samples** illustrating the various features provided by JORAM.

JORAM platform and client classes may be downloaded as packages or can be checked out from a CVS repository.

JORAM communication generally relies on TCP. But specific clients (such as kJORAM clients) rely on the **SOAP** protocol for communicating with the messaging server. For those clients, it is required to install Apache (<http://www.apache.org>) servlet container, **Tomcat**. How to configure *Tomcat* is explained by the last section of this document.

2. Getting JORAM

2.1. Installing a package

The packages are downloadable from the following location:
http://forge.objectweb.org/project/showfiles.php?group_id=4.

For release **x.y.z**, the following tar files are provided:

- **joram-x.y.z.tgz**, including the client and server libraries, as well as the graphical tool library, and the samples sources.
- **joram-x.y.z-src.tgz**, including the client and server sources, as well as the graphical tool sources, and the samples sources.
- **kjoram-x.y.z.tgz**, including the J2ME client libraries, the J2ME samples sources. **Server libraries are not provided.**
- **kjoram-x.y.z-src.tgz**, including the J2ME client sources, as well as the J2ME samples sources. **Server sources are not provided.**

A package is expanded by UNIX users with the *gunzip* and *tar* commands; Windows developers can use the *Winzip* utility.

2.2. Getting JORAM from CVS

JORAM CVS page is located at: http://forge.objectweb.org/cvs/?group_id=4.

For a JORAM checkout, type:

```

cvs
-d:pserver:anonymous@cvs.joram.debian-sf.objectweb.org:/cvsroot/joram
login

```

When prompted for a password, simply press the **Enter** key.

The module to extract is **joram**:

```

cvs
-d:pserver:anonymous@cvs.joram.debian-sf.objectweb.org:/cvsroot/joram
co joram

```

2.3. Directory structure

JORAM binary distribution

The distribution is expanded in a `joram-x.y.z/` directory. It includes the following directories:

- `lib/`
- `licenses/`

- samples/
 - bin/...
 - config/
 - src/
 - joram/...

JORAM sources distribution

The distribution is expanded in a `joram-x.y.z-src/` directory. It includes the following directories:

- lib/
- licenses/
- samples/
 - bin/...
 - config/
 - src/
 - joram/...
- src/
 - fr/...
 - org/...

kJORAM binary distribution

The distribution is expanded in a `kjoram-x.y.z/` directory. It includes the following directories:

- lib/
- licenses/
- samples/
 - src/
 - kjoram/...

kJORAM sources distribution

The distribution is expanded in a `kjoram-x.y.z-src/` directory. It includes the following directories:

- lib/
- licenses/
- samples/
 - src/
 - kjoram/...
- src/
 - com/...

JORAM CVS module

Getting JORAM from CVS creates a `joram/` directory with the following structure:

- lib/
- licenses/
- samples/
 - bin/...
 - config/
 - src/
 - joram/...
 - kjoram/...
- src/
 - com/...
 - fr/...
 - org/...

2.4. Directories description

lib/ directory

Contains the libraries needed for compiling the distribution, and running the samples.

- Included in all JORAM distributions **except kJORAM packages**:
 - `activation.jar`, JavaBeans activation framework;
 - `jakarta-regexp-1.2.jar`, Jakarta's regular expression parser;
 - `JCup.jar`, javaCup Java parser generator;
 - `jms.jar`, the JMS API;
 - `jndi.jar`, the JNDI API;
 - `joram.jar`, JORAM client library, **binary package only**;
 - `joramgui.jar`, JORAM administration tool library, **binary package only**;
 - `jta.jar`, the JTA API;
 - `log4j.jar`, Jakarta's logger;
 - `mail.jar`, JavaMail API;
 - `mom.jar`, JORAM server library, **binary package only**;
 - `ow_monolog.jar`, ObjectWeb's logger API and its wrapper for log4j;
 - `soap.jar`, Apache's SOAP implementation;
 - `soap.war`, Apache's SOAP web archive file;
 - `xerces.jar`, Apache's XML parser.
- Included in the **kJORAM packages** and in CVS:
 - `kjoram.jar`, kJORAM client library, **binary package only**;
 - `kxml.jar`, Enhydra's kXML implementation;
 - `midpapi.jar`, Sun's MIDP API.

licenses/ directory

Contains the LGPL header displayed on top of each source file, as well as the licences of the external softwares provided in the distribution.

samples/bin directory

Contains UNIX and Windows scripts for launching JORAM servers and clients (how to use them is explained in the samples document).

samples/config directory

Contains configuration files needed for running JORAM samples:

- `a3config.dtd`, the DTD for server configuration;
- `a3debug.cfg`, the logger configuration file;
- `centralized_a3servers.xml`, a configuration file for a centralized server architecture;
- `distributed_a3servers.xml`, a configuration file for a distributed servers architecture;
- `soap_a3servers.xml`, the configuration file for the SOAP samples server architecture;
- `soap_jndi.properties`, the configuration file for JNDI's SOAP clients;
- `tcp_jndi.properties`, a configuration file for JNDI's TCP clients.

samples/src/joram directory

Contains the sources of the JORAM samples.

samples/src/kjoram directory

Contains the sources of the kJORAM samples as well as utility files for running the samples on a Pocket PC device.

src/com directory

Contains the sources of the kJORAM client.

src/fr directory

Contains the sources of JORAM server and client.

src/org/objectweb/joram/tools/admin directory

Contains the sources of JORAM graphical administration tool.

3. Compiling

JORAM distribution is ready for compiling with Apache **Ant** utility. *Ant* can be downloaded from <http://jakarta.apache.org/ant/>. Documentation is available at the same location.

3.1. Compiling the CVS tree

When working with CVS, it is required to compile JORAM sources. In the `src/` directory, 3 build files are provided:

- `joram.xml` specifies the targets for compiling and shipping the MOM classes, the Joram client classes and the graphical admin tool classes;
- `kjoram.xml` specifies the targets for compiling and shipping the kJoram client classes;
- `build.xml` specifies the main targets for convenience.

The available *Ant* targets can be listed by typing:

```
ant -projecthelp
ant -buildfile joram.xml -projecthelp
ant -buildfile kjoram.xml -projecthelp
```

Generating the javadoc

Simply type:

```
ant javadoc (equivalent to ant -buildfile joram.xml javadoc)
```

This creates the `classes/` and `javadoc/` directories when generating the doc is requested for the first time or after a clean. The `classes/` directory stays empty, `javadoc/` holds the doc.

Shipping Joram

The best way to build Joram is by using the default shipping target.

```
ant (equivalent to ant -buildfile joram.xml ship.joram)
```

This creates the `classes/` and `javadoc/` directories when compiling is requested for the first time or after a clean. The `classes/` directory holds the compiled classes, the `joram.jar` and `mom.jar` libraries are created in the `lib/` directory.

Shipping kJoram

Building kJoram requires to use a 1.3 jdk, and to modify the compiler in the `build.properties` files as "classic" instead of "modern". Then, type:

```
ant ship.kjoram (equivalent to ant -buildfile kjoram.xml ship.kjoram)
```

This creates the `classes/` and `javadoc/` directories when compiling is requested for the first time or after a clean. The `classes/` directory holds the compiled classes, the `kjoram.jar` library is created in the `lib/` directory.

Compiling the administration tool

Compiling the administration tool requires to use jdk 1.4. It is done by typing:

```
ant ship.gui (equivalent to ant -buildfile joram.xml ship.gui)
```

This creates the `classes/` and `javadoc/` directories when compiling is requested for the first time or after a clean. The generated `joramgui.jar` library is created in the `lib/` directory, the `classes/` directory holds the compiled classes.

Cleaning

To remove the generated classes and libraries:

```
ant clean (equivalent to ant -buildfile joram.xml clean and ant -buildfile kjoram.xml clean)
```

This removes the `classes/` and `javadoc/` directories if they exist, the `joram.jar`, `kjoram.jar`, `joramgui.jar` and `mom.jar` libraries from the `lib/` directory if they exist.

3.2. Compiling the sources package

In the `src/` directory, a single `build.xml` build file is provided and specifies the needed targets.

The available *Ant* targets can be listed by typing:

```
ant -projecthelp
```

Generating the javadoc

Simply type:

```
ant javadoc
```

This creates the `classes/` and `javadoc/` directories when generating the doc is requested for the first time or after a clean. The `classes/` directory is empty, `javadoc/` holds the doc.

Shipping Joram

The default shipping target is:

```
ant (equivalent to ant ship.joram)
```

This creates the `classes/` and `javadoc/` directories when compiling is requested for the first time or after a clean. The `classes/` directory holds the compiled classes, the `joram.jar` and `mom.jar` libraries are created in the `lib/` directory.

Compiling the administration tool

Compiling the administration tool requires to use jdk 1.4. It is done by typing:

```
ant ship.gui
```

This creates the `classes/` and `javadoc/` directories when compiling is requested for the first time or after a clean. The generated `joramgui.jar` library is created in the `lib/` directory, the `classes/` directory holds the compiled classes.

Cleaning

To remove the generated classes and libraries:

```
ant clean
```

This removes the `classes/` and `javadoc/` directories if they exist, the `joram.jar`, `joramgui.jar` and `mom.jar` libraries from the `lib/` directory if they exist.

3.3. *Compiling JORAM samples*

The samples in the JORAM packages and in the CVS distribution need to be compiled. Under the `samples/src/joram` directory, simply type:

```
ant
```

This creates a `samples/classes/joram/` directory holding the compiled classes. For removing this directory, type:

```
ant clean
```

How to run the samples is fully described in the samples documentation.

3.4. *Compiling kJORAM samples*

The samples in the kJORAM packages and in the CVS distribution need to be compiled. Under the `samples/src/kjoram` directory, simply type:

```
ant
```

This creates a `samples/classes/kjoram/` directory holding the compiled classes. For removing this directory, type:

```
ant clean
```

As kJORAM sources, the samples are compiled with **target 1.1** for J2ME compliance; how to run them is fully described in the samples documentation.

4. Shipping JORAM

This section describes which libraries and configuration files are needed for running a JORAM server, the graphical administration and monitoring tool, a JORAM client, and a kJORAM client.

4.1. JORAM server

Libraries

- jakarta-regexp-1.2.jar,
- JCup.jar,
- log4j.jar,
- mom.jar,
- ow_monolog.jar,
- xerces.jar.

Configuration files

- a3debug.cfg,
- a3servers.xml.

4.2. Graphical tool

Libraries

- JCup.jar,
- jms.jar,
- jndi.jar,
- joram.jar,
- joramgui.jar
- log4j.jar,
- ow_monolog.jar.

Configuration files

- a3debug.cfg,
- jndi.properties.

4.3. *JORAM client*

Libraries

- `activation.jar` (needed when running SOAP clients),
- `JCup.jar`,
- `jms.jar`,
- `jndi.jar`,
- `joram.jar`,
- `jta.jar`,
- `log4j.jar`,
- `mail.jar` (needed when running SOAP clients),
- `ow_monolog.jar`,
- `soap.jar` (needed when compiling and running SOAP clients).

Configuration files

- `a3debug.cfg`,
- `jndi.properties`.

4.4. *kJORAM client*

Libraries

- `kjoram.jar`,
- `kxml.jar`.

5. Configuring Tomcat

The JORAM platform includes a SOAP proxy, accepting connection requests from SOAP clients. This proxy requires the installation and the setting of Apache servlet container, **Tomcat**.

JORAM has been successfully tested with *tomcat-3-3*, *tomcat-4-0* and *tomcat-4-1*. This section describes how those versions should be configured.

Getting Tomcat

Tomcat can be downloaded from <http://jakarta.apache.org/tomcat/>. Documentation is available at the same location.

Needed resources

Libraries and configuration files must be added in the *Tomcat* environment. The jar and war files must be taken from the `lib/` directory of your JORAM installation. The configuration files must be taken from the `samples/config` directory of your JORAM installation.

Configuring Tomcat 3.3

- Files to modify in the `bin/` directory:
 - in `tomcat.bat` or `tomcat.sh`: add the `conf/` directory in the **classpath** building.
- Libraries to add in the `lib/common` directory:
 - `activation.jar`,
 - `jakarta-regexp-1.2.jar`,
 - `JCup.jar`,
 - `jms.jar`,
 - `joram.jar`,
 - `log4j.jar`,
 - `mail.jar`,
 - `mom.jar`,
 - `ow_monolog.jar`,
 - `xerces.jar`.
- Files to add in the `conf/` directory :
 - `a3debug.cfg`,
 - `soap_a3servers.xml` renamed `a3servers.xml`.
- File to add in the `webapps/` directory:
 - `soap.war`.

Configuring *Tomcat 4.0*

- Files to modify in the `bin/` directory:
 - in `setclasspath.bat` or `setclasspath.sh`: add the `conf/` directory in the **classpath** building.
- Libraries to add in the `lib/common` directory:
 - `jakarta-regexp-1.2.jar`,
 - `JCup.jar`,
 - `jms.jar`,
 - `joram.jar`,
 - `log4j.jar`,
 - `mom.jar`,
 - `ow_monolog.jar`,
 - `xerces.jar` (replace the one provided with Tomcat).
- Files to add in the `conf/` directory :
 - `a3debug.cfg`,
 - `soap_a3servers.xml` renamed `a3servers.xml`.
- File to add in the `webapps/` directory:
 - `soap.war`.

Configuring *Tomcat 4.1*

- Files to modify in the `bin/` directory:
 - in `setclasspath.bat` or `setclasspath.sh`: add the `conf/` directory in the **classpath** building.
- Libraries to add in the `common/lib` directory:
 - `jakarta-regexp-1.2.jar`,
 - `JCup.jar`,
 - `jms.jar`,
 - `joram.jar`,
 - `log4j.jar`,
 - `mom.jar`,
 - `ow_monolog.jar`.
- Files to add in the `conf/` directory :
 - `a3debug.cfg`,
 - `soap_a3servers.xml` renamed `a3servers.xml`.
- File to add in the `webapps/` directory:
 - `soap.war`.