Leading Open Source Middleware

JOnAS Versions Migration Guide

JOnAS Team (Philippe Coq)

- March 2009 -

Copyright © OW2 Consortium 2008-2009

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license,visit http://creativecommons.org/licenses/by-sa/2.0/deed.en or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Table of Contents

Chapter 1. Migrating from JOnAS 4.x to JOnAS 5

JOnAS has been completely rewritten in order to leverage the OSGi™ technology.

People who were familiar with the JOnAS world will not be amazed. However, they should be aware of some important changes that must be taken into account.

1.1. Before migration

Before begining the migration process, you must be aware of the versions of the Java EE API supported by JOnAS 5.

JOnAS 4 is Java EE v1.4 compliant whereas. JOnAS 5 is Java EE 5 compliant. This means that the supported versions of most APIs have changed.

Here is a small table describing the differences:

1.1.1. API versions of the Java EE components

API	JOnAS 4.x	JOnAS 5.x
JDK	1.4	1.5
JNDI	1.2	1.2
EJB	2.1	3.0
Servlet	2.4	2.5
JSP	2.0	2.1
JMS	1.1	1.1
Connector Architecture	1.5	1.5
JDBC	3.0	3.0
JavaMail	1.3.1	1.4
JavaBeans Activation Framework	1.0	1.1.1
JTA	1.0.1B	1.1
JTS	1.0	1.0
JAX-RPC	1.1	1.1
SAAJ	1.2	1.3
JAXR	1.0	1.0
JWSDL	1.0	1.0
JAXP	1.2	1.3
J2EE_Deployment	1.1	1.1
J2EE_Mgmt	1.0	1.0
JACC	1.0	1.1
JAAS	1.0	1.0
JAX-WS	n/a	2.1
JAXB	n/a	2.1

API	JOnAS 4.x	JOnAS 5.x
JSF	n/a	1.2
JSTL	n/a	1.2
JPA	n/a	1.0

1.1.2. Prerequisites

A JDK 5 is required to build JOnAS 5 but JDK 6[1]may be used for running JOnAS 5.

1.2. Upgrading An Application Environment from 4.x to 5.x

1.2.1. Application environment definition

Usually, an application environment is composed of one JONAS_ROOT (JOnAS distribution) and one or more JONAS_BASE depending on the number of JOnAS instances needed for the application.

In JOnAS 4.x JONAS_BASE contains :

- Java EE applications to be deployed in the new environment:

  An application consists of several deployable archives: ear, ejbjar, war + some associated connectors packaged in rar archives.

- Configuration files:

  All configuration files are located under `$JONAS_BASE/conf`

- External libraries:

  Some libraries may be installed under:

  - `$JONAS_BASE/lib/apps`

    (applicative libraries)

  - `$JONAS_BASE/lib/ext`
  - `$JONAS_BASE/lib/commons`
  - `$JONAS_BASE/lib/tools`

    (libraries whoses classes are made available to all JOnAS services)

For a successful upgrade you must be aware of the JOnAS changes in :

- the class loader hierarchy see Understanding class loader hierarchy [j2eeprogrammerguide.html#j2ee.pg.classloader]for a complete description of the classloader mechanism.
- deployment process, due to a new deployment architecture
- configuration files
- third party libraries processing

[1]for those using SUN JDK6 the parameter -Dsun.lang.ClassLoader.allowArraySyntax=true may be required

To upgrade an application environment you must first create a new application environment and then customize it according to the old environment.

1.2.1.1. Creating a new Application environment

To create a new application environnement you must create a new JOnAS_BASE.

1.2.1.1.1. JONAS_BASE creation

1. To create a JONAS_BASE template from scratch :

   Unix

   ```
   export JONAS_BASE=~/my_jonas_base
   cd $JONAS_ROOT/templates/newjb
   ant -f build-jb.xml create_jonas_base
   ```

   Windows

   ```
   set JONAS_BASE=my_jonas_base
   cd %JONAS_ROOT%/templates/newjb
   ant -f build-jb.xml create_jonas_base
   ```

   This will copy all the required files and create all the needed directories.

2. Another way to create a JONAS_BASE template from scratch :

   `$JONAS_ROOT/bin` must be set in the system path:

   Unix

   ```
   export JONAS_BASE=~/my_jonas_base
   newjb
   ```

   Windows

   ```
   set JONAS_BASE=my_jonas_base
   newjb
   ```

   The JONAS_BASE content created with the **newjb** command is well suited to run the JOnAS JEE conformance test suite and the example applications without any additional configuration.

   In order to customize a JONAS_BASE with specific property values (port numbers, services, protocols etc...), you must edit the `$JONAS_ROOT/templates/newjb/build-jb.properties` file or `$HOME/jb.config/conf/jonas-newjb.properties` file before running **newjb**.

   For further customization that cannot be performed by **newjb** you should modify the generated files in `$JONAS_BASE/conf`. For more information see the description of the newjb command in Commands Reference Guide [./command_guide.html#commands.newjb].

Note

When running newjb based on information located in your home directory ($HOME/jb.config/conf/jonas-newjb.properties), you will probably have to review this file and upgrade it to conform to $JONAS_ROOT/build-jb.properties syntax. For example, jonas services names have changed between JOnAS version 4 and version 5.

1.2.2. New deployment architecture

With JOnAS 5.x, the following deployment directories have been removed :

- `apps`
- `apps/autoload`
- `clients`
- `ejbjars`
- `ejbjars/autoload`
- `rars`
- `rars/autoload`
- `webapps`
- `webapps/autoload`

The above directories no longer exist in JONAS_BASE and have been replaced by a single directory:

- `deploy`

To deploy a Java EE archive in JOnAS 5.x, the only thing required is to put it in the `$JONAS_BASE/deploy directory`. This directory is periodically scanned in order to deploy new archives. [2]

The deployment policy has changed in JOnAS 5 since at server starting time the deployment order of the archives found in $JONAS_BASE/deploy is the following:

1. Deployment plan repositories
2. OSGi™ bundles
3. RAR archives
4. Deployment plan resources
5. EJB archives
6. WAR archives
7. EAR archives

Note

For each category, file names are chosen in alphabetical order

Here a basic *how to* that explains how to create a very simple deployment plan.

1.2.2.1. Deployment plan sample example

This example shows how to control the deployment of two archives which are located on the file system

Myear1.ear and Myear2.ear are located in local directory `/home/elsewhere`.

- Add a new repository element in $JONAS_BASE/conf/initial-repositories.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
```

[2] The periodic scan can be disabled by setting the jonas.development property to false in `JONAS_BASE/conf/jonas.properties`

```
<repositories
  xmlns="http://jonas.ow2.org/ns/deployment-plan/repositories/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="repositories-1.0.xsd">

  <!-- Add new repositories here -->
  <repository id="MyLocalRepository">
    <type>url</type>
    <url>file:/home/elsewhere</url>
  </repository>
  <!-- Default repositories -->
  <repository id="maven2-central">
    <type>maven2</type>
    <url>http://repo1.maven.org/maven2/</url>
  </repository>
  <repository id="maven2-ow2-release">
    <type>maven2</type>
    <url>http://maven.ow2.org/maven2/</url>
  </repository>
  <repository id="maven2-ow2-snapshot">
    <type>maven2</type>
    <url>http://maven.ow2.org/maven2-snapshot/</url>
  </repository>

</repositories>
```

- create a new deployment plan $JONAS_BASE/deploy/MyDeploymentplan.xml

```
  <?xml version="1.0" encoding="UTF-8"?>
    <deployment-plan xmlns="http://jonas.ow2.org/ns/deployment-plan/1.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:url="http://jonas.ow2.org/ns/deployment-plan/url/1.0">

    <deployment id="foo1" xsi:type="url:url-deploymentType">
      <url:resource>Myear1.ear</url:resource>
    </deployment>

    <deployment id="foo2" xsi:type="url:url-deploymentType">
      <url:resource>Myear2.ear</url:resource>
    </deployment>

  </deployment-plan>
```

  As the deployment plan is under $JONAS_BASE/deploy/ it will be taken into account at server starting time.

For more information see the depmonitor service configuration in the JOnAS Configuration Guide. [configuration_guide.html#services.depmonitor.config]

1.2.3. Script files

The path of the script command files has been changed, and is `$JONAS_ROOT/bin` instead of `$JONAS_ROOT/bin/unix` and `$JONAS_ROOT/bin/nt`.

1.2.4. Configuration files

JOnAS configuration files have changed. Here we will focus on the changes that may impact the upgrade process.

1.2.4.1. carol.properties file

jeremie protocol is no longer supported.

New properties have been added to configure cmi clusters. Note that **cmi** is not anymore a protocol but a JOnAS service used for clustering.

For applications that uses a carol.properties file on the client side the best practice is to start from $JONAS_ROOT/template/conf/conf/carol.properties and to customize it (see the JOnAS Configuration Guide). [configuration_guide.html#config.protocol] .

1.2.4.2. clusterd.properties file

With JOnAS 5 the cluster deamon can start the discovery service.

1.2.4.3. context.xml, server.xml, web.xml

JOnAS 5 now embeds Tomcat 6 instead of Tomcat 5.

`context.xml, server.xml and web.xml` configuration files are replaced by `tomcat6-context.xml`, `tomcat6-server.xml` and `tomcat6-web.xml`.

See Apache Tomcat 6.0 Documentation [http://tomcat.apache.org/tomcat-6.0-doc/index.html] for more information.

1.2.4.4. jaas.config

No new properties in this file. The only difference is the name of the package for the LoginModule classes.

For applications that uses a jaas.config on the client side the best practice is to start from $JONAS_ROOT/template/conf/conf/jaas.config and to customize it.

1.2.4.5. jetty5-webdefault.xml & jetty5.xml

These configuration files are now named `jetty6-web.xml and jetty6.xml.`

For more information see Jetty Documentation [http://docs.codehaus.org/display/JETTY/Jetty+Documentation]

1.2.4.6. jgroups-cmi.xml & jgroups-ha.xml

JOnAS 5 now uses a newer version of JGroups. The JGroups Protocol Stack configuration files format is different. Therefore customizations made in the old `jgroups-cmi.xml` and `jgroups-ha.xml` files must be updated to the new formats.

1.2.4.7. jonas.properties

The `jonas.properties` file has changed significantly . Some changes do not impact the upgrade process at all (package name changes), while others must be studied carefully.

In the following, we focus on the parts of the `jonas.properties` file that may be impacted during the upgrade process.

1.2.4.7.1. jonas.services property

thread and **jms** services have been suppressed (see Section 1.2.4.7.3, “jms service configuration” below).

ejb service has been renamed to **ejb2** service.

ws service has been renamed to **jaxrpc** service.

wm, **wc**, **ha**, **ejb3**, **versioning**, **depmonitor**, **cmi**,**resourcemonitor**,**jaxws** and **smartclient** are new services in JOnAS 5

With JOnAS 5, the default jonas.services properties are set to:

```
jonas.services    jtm,db,dbm,security,resource,ejb3,jaxws,web,ear,depmonitor
```

(in order to learn more about these services see the JOnAS configuration_guide [configuration_guide.html#config.services])

1.2.4.7.2. ejb2 service configuration

ejb2 service replaces the previous **ejb** service.

Configuration of worker threads has been moved from the **ejb** service configuration to the Work Manager (**wm**) service configuration.

See the Work Manager configuration part in Section 1.2.4.7.4, “resource service configuration”

1.2.4.7.3. jms service configuration

The JORAM resource adapter must be used in replacement of the **jms** service.

In order to use JMS via the JORAM resource adapter

1. The **resource** service must be set in the jonas.services property
2. `joram.xml` must be deployed (it must be located under `$JONAS_BASE/deploy`)
3. JORAM port must be configured in `the $JONAS_BASE/conf/joramAdmin.xml` file
4. JMS destinations must be defined in `the $JONAS_BASE/conf/joramAdmin.xml` file

If for tuning purposes, the size and/or the max size of the worker thread pool used for the message driven beans has been set (jonas.service.ejb.minworkthreads, jonas.service.ejb.maxworkthreads properties) you must update the equivalent settings in the Work Manager configuration (see below).

1.2.4.7.4. resource service configuration

Configuration of worker threads has been moved from **resource** service configuration to Work Manager (**wm**) service configuration.

Work Manager configuration:

```
###################### JOnAS WorkManager service configuration
#
#  Set the name of the implementation class of the wm service
jonas.service.wm.class    org.ow2.jonas.workmanager.internal.JOnASWorkManagerService

#  Set the size of the worker thread pool
jonas.service.wm.minworkthreads   3                                 1

#  Set the maximun size of the worker thread pool
jonas.service.wm.maxworkthreads   80                                2

#  Set the max # of seconds that a thread will wait for work
#  This is used to shrink the worker thread pool back to minimum
jonas.service.wm.threadwaittimeout 60                               3
```

1 Defines the minimum size of the Thread pool
2 Defines the maximum size of the Thread pool
3 Defines the maximum time (in seconds) that a worker Thread should wait before execution

1.2.4.7.5. discovery service configuration

The property indicating that the current instance is (or is not) a master server is no longer a **discovery** service property, but instead, a `jonas.properties` global property

```
jonas.master false
```

The **discovery** service supports two implementations. One of them uses JGroups, the other one uses multicast. Multicast is the default..

For more information about configuring discovery service see discovery service configuration. [configuration_guide.html#services.discovery.config]

1.2.4.8. trace.properties & traceclient.properties

Logger names have changed because of changes in internal JOnAS packages name.

1.2.5. libraries

There are several ways to use libraries in portable Java EE applications :

- placing the library JAR file into the WEB-INF/lib directory of the WAR file,
- using the Class-Path attribute in the manifest file to reference one or more library JAR files included in the EAR file.
- using the Extension-List attribute in the manifest file to reference one or more library JAR files that are not bundled in the EAR file, but are installed in the lib/ext directory of the Java Runtime Environment (JRE)

If the application to migrate follows one of these policies it can be used with JOnAS 5 without changes.

There is no guarantee that a library installed in JOnAS_BASE/lib/ext and referenced via the Extension-List attribute in the manifest file will be accessible to your application.

It is worth knowing that tomcat 6, embedded in JOnAS 5, tries to resolve extension dependencies for web applications before deploying them, and that it uses the system property catalina.ext.dirs to try and locate libraries.

The JDBC driver can be installed under $JONAS_BASE/lib/ext or packaged into an OSGi™ bundle, we recommend the latest way.

1.2.6. Using the Management bean MEJB

1.2.6.1. Access to the Management EJB Component

Access to the MEJB provided by the JOnAS 5 distribution is now role secured. The client application that wants to access it must have one of the following roles:

mejb-user for accessing to the read-only operation of the MEJB

mejb-admin role allowing access for all operation of the MEJB

Unauthorized access is forbidden

This snippet includes all of the elements needed to specify security roles in the `web.xml` file of a web application client of MEJB

```
<security-constraint>
<web-resource-collection>
  <web-resource-name>Protected Area</web-resource-name>
  <!-- Define the context-relative URL(s) to be protected -->
  <url-pattern>/*</url-pattern>
  <!-- If you list http methods, only those methods are protected -->
  <http-method>DELETE</http-method>
  <http-method>GET</http-method>
  <http-method>POST</http-method>
  <http-method>PUT</http-method>
</web-resource-collection>
<auth-constraint>
  <!-- Anyone with one of the listed roles may access this area -->
  <role-name>mejb-admin</role-name>
</auth-constraint>
```

```
</security-constraint>

<!-- Default login configuration uses BASIC authentication -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>JOnAS Realm</realm-name>
</login-config>

<!-- Security roles referenced by this web application -->
<security-role>
  <role-name>mejb-admin</role-name>
</security-role>
```

1.3. Changes that can affect application code

If the application to port was pointing to JOnAS java internal classes with **org.objectweb.jonas.xxx** packages, all references to these packages must be replaced by references to **org.ow2.jonas.xxx** package.

For example if the application is using the authentication mechanism implemented by the class **org.objectweb.jonas.security.realm.web.catalina55.JACC** in JOnAS 4 this mechanism is now implemented in JOnAS5 by **org.ow2.jonas.web.tomcat6.security.Realm.**

org.objectweb.jonas.security.realm.web.jetty50.Standard must be changed by **org.ow2.jonas.web.jetty6.security.Realm**

org.objectweb.jonas.security.auth.callback.LoginCallbackHandler must be changed by **org.ow2.jonas.security.auth.callback.LoginCallbackHandler**

org.objectweb.jonas.security.auth.spi.JResourceLoginModule by **org.ow2.jonas.security.auth.spi.JResourceLoginModule**

org.objectweb.jonas.security.auth.spi.ClientLoginModule by **org.ow2.jonas.security.auth.spi.ClientLoginModule**