Leading Open Source Middleware

J2EE Connector Programmer's Guide

JOnAS Team (Eric HARDESTY)

- March 2009 -

Table of Contents

This guide is provided for advanced JOnAS users concerned with EAI (Enterprise Application Integration) and using the J2EE Connector Architecture principles (refer to Section A.2, “Connector Architecture Principles” for an introduction to the connectors). The target audience for this guide is the Resource Adapter deployer and programmer. It describes the JOnAS specific deployment file (jonas-ra.xml) and the sample code to access deployed RARs.

Chapter 1. Using a J2EE Connector

1.1. Principles

Resource Adapters are packaged for deployment in a standard Java programming language Archive file called a rar file (Resource ARchive), which is described in the J2EE Connector Architecture specification.

The standard method for creating the jonas-ra.xml file is to use the RAConfig command [command_guide.html#commands.raconfig]

1.2. Defining the JOnAS Connector Deployment Descriptor

The jonas-ra.xml contains JOnAS specific information describing deployment information, logging, pooling, jdbc connections, and RAR config property values.

1.2.1. Deployment Tags

jndiname (Required)	Name the RAR will be registered as. This value will be used in the resource-ref section of an EJB.
rarlink	Jndiname of a base RAR file. Useful for deploying multiple connection factories without having to deploy the complete RAR file again. When this is used, the only entry in RAR is a META-INF/jonas-ra.xml.
native-lib	Directory where additional files in the RAR should be deployed.

1.2.2. Logging Tags

log-enabled	Determines if logging should be enabled for the RAR.
log-topic	Log topic to use for the PrintWriter logger, which allows a separate handler for each deployed RAR.

1.2.3. Pooling Tags

pool-init	Initial size of the managed connection pool.
pool-min	Minimum size of the managed connection pool.
pool-max	Maximum size of the managed connection pool. Value of -1 is unlimited.
pool-max-age	Maximum number of milliseconds to keep the managed connection in the pool. Value of 0 is an unlimited amount of time.
pstmt-max	Maximum number of PreparedStatements per managed connection in the pool. Only needed with the JDBC RA of JOnAS or another database vendor's RAR. Value of 0 is unlimited and -1 disables the cache.
pstmt-cache-policy	Prepared statement cache policy. Possible values are List or Map.

1.2.4. JDBC Connection Tags

Note

Only valid with a Connection implementation of java.sql.Connection.

jdbc-check-level	Level of checking that will be done for the jdbc connection. Values are 0 for no checking, 1 to validate that the connection is not closed before returning it, and greater than 1 to send the jdbc-test-statement.
jdbc-test-statement	Test SQL statement sent on the connection if the jdbc-check-level is set accordingly.

1.2.5. Config Property Value Tags

Each entry must correspond to the config-property specified in the ra.xml of the RAR file. The default values specified in the ra.xml will be loaded first and any values set in the jonas-ra.xml will override the specified defaults.

1.2.6. Deployment Descriptor Examples

The following portion of a jonas-ra.xml file shows the linking to a base RAR file named BaseRar. All properties from the base RAR will be inherited and any values given in this jonas-ra.xml will override the other values.

```
  <jonas-resource>
    <jndiname>rar1</jndiname>
    <rarlink>BaseRar</rarlink>
    <native-lib>nativelib</native-lib>
    <log-enabled>false</log-enabled>
    <log-topic>com.xxx.rar1</log-topic>
    <jonas-config-property>
      <jonas-config-property-name>ip</jonas-config-property-name>
      <jonas-config-property-value>www.xxx.com</jonas-config-property-value>
    </jonas-config-property>
    .
    .
  </jonas-resource>
```

The following portion of a jonas-ra.xml file shows the configuration of a jdbc rar file.

```
  <jonas-resource>
    <jndiname>jdbc1</jndiname>
    <rarlink></rarlink>
    <native-lib>nativelib</native-lib>
    <log-enabled>false</log-enabled>
    <log-topic>com.xxx.jdbc1</log-topic>
    <pool-params>
      <pool-init>0</pool-init>
      <pool-min>0</pool-min>
      <pool-max>100</pool-max>
      <pool-max-age>0</pool-max-age>
      <pstmt-max>20</pstmt-max>
      <pstmt-cache-policy>Map</pstmt-cache-policy>
    </pool-params>
    <jdbc-conn-params>
      <jdbc_check-level>2</jdbc_check-level>
      <jdbc-test-statement>select 1</jdbc-test-statement>
    </jdbc-conn-params>
    <jonas-config-property>
      <jonas-config-property-name>url</jonas-config-property-name>
      <jonas-config-property-value>jdbc:oracle:thin:@test:1521:DB1</jonas-config-property-
value>
    </jonas-config-property>
    .
    .
  </jonas-resource>
```

1.3. Resource Adapter (RAR) Packaging

Resource Adapters are packaged for deployment in a standard Java programming language Archive file called an RAR file (Resource Adapter ARchive). This file can contain the following:

Resource Adapters' deployment descriptor	The RAR file must contain the deployment descriptors, which are made up of: • The standard xml deployment descriptor, in the format defined in the J2EE 1.4 specification. Refer to http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd [http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd] . This deployment descriptor must be stored with the name META-INF/ra.xml in the RAR file. • The JOnAS-specific XML deployment descriptor in the format defined in http://jonas.objectweb.org/ns/jonas-connector_5_0.xsd [http://jonas.ow2.org/ns/jonas-connector_5_0.xsd] This JOnAS deployment descriptor must be stored with the name META-INF/jonas-ra.xml in the RAR file.
Resource adapter components (jar)	One or more jars which contain the java interfaces, implementation, and utility classes required by the resource adapter.
Platform-specific native libraries	One or more native libraries used by the resource adapter
Misc	One or more html, image files, or locale files used by the resource adapter.

Before deploying an RAR file, the JOnAS-specific XML must be configured and added. Refer to the RAConfig command [command_guide.html#commands.raconfig] for information.

1.4. Use and Deployment of a Resource Adapter

Accessing Resource Adapter involves the following steps:

1. The bean provider must specify the connection factory requirements by declaring a resource manager connection factory reference in its EJB deployment descriptor. For example:

```
<resource-ref>
  <res-ref-name>eis/MyEIS</res-ref-name>
  <res-type>javax.resource.cci.ConnectionFactory</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

   The mapping to the actual JNDI name of the connection factory (here `adapt_1`) is done in the JOnAS-specific deployment descriptor with the following element:

```
<jonas-resource>
  <res-ref-name>eis/MyEIS</res-ref-name>
  <jndi-name>adapt_1</jndi-name>
</jonas-resource>
```

   This means that the bean programmer will have access to a connection factory instance using the JNDI interface via the java:comp/env/eis/MyEIS name:

```
// obtain the initial JNDI naming context
Context inictx = new InitialContext();

// perform JNDI lookup to obtain the connection factory
```

```
javax.resource.cci.ConnectionFactory cxf =
        (javax.resource.cci.ConnectionFactory)
             inictx .lookup("java:comp/env/eis/MyEIS");
```

The bean programmer can then get a connection by calling the method `getConnection` on the connection factory .

```
javax.resource.cci.Connection cx = cxf.getConnection();
```

The returned connection instance represents an application-level handle to a physical connection for accessing the underlying EIS. After finishing with the connection, it must be closed using the `close` method on the `Connection` interface:

```
cx.close();
```

2. The resource adapter must be deployed before being used by the application. Deploying the resource adapter requires the following:

   Build a JOnAS-specific resource adapter configuration file that will be included in the resource adapter. This jonas-ra XML file is used to configure the resource adapter in the operational environment and reflects the values of all properties declared in the deployment descriptor for the resource adapter, plus additional JOnAS-specific configuration properties. JOnAS provides a deployment tool RAConfig [command_guide.html#commands.raconfig] that is capable of building this XML file from an RA deployment descriptor inside an RAR file. Example:

   ```
   RAConfig -path . -j adap_1 ra
   ```

   These properties may be specific for each resource adapter and its underlying EIS. They are used to configure the resource adapter via its `managedConnectionFactory` class. It is mandatory that this class provide getter and setter method for each of its supported properties (as it is required in the Connector Architecture specification).

   After configuring the jonas-ra.xml file created above, it can be added to the resource adapter by executing the following:

   ```
   RAConfig -u jonas-ra.xml ra
   ```

   This will add the xml file to the ra.rar file, which is now ready for deployment.

3. The JOnAS resource service must be configured and started at JOnAS launching time: In the `jonas.properties` file:

   - Verify that the name `resource` is included in the `jonas.services` property.
   - Use one of the following methods to deploy an RAR file:
     - The names of the resource adapter files (the '.rar' suffix is optional) must be added in the list of Resource Adapters to be used in the `jonas.service.resource.resources` property. If the '.rar' suffix is not used on the property, it will be used when trying to allocate the specified Resource Adapter.

       ```
       jonas.service.resource.resources  MyEIS.rar, MyEIS1
       ```

- Place the RAR file in the connectors autoload directory of $JONAS_BASE, default value is $JONAS_BASE/rars/autoload. Note that it may be different if `jonas.service.resource.autoload` in jonas.properties is configured differently.
- Add the RAR via the `jonas admin -a xxx.rar` command.
- Add the RAR via the JonasAdmin console.

Chapter 2. Exemples of Resource Adapters

2.1. Configuring JDBC Resource Adapters

Connection of an J2EE application to databases is done through JDBC Resource Adapters (JDBC RA).

Such Resource Adapters are deployed via the **resource** service as seen in Section A.1, “resource service configuration”.

For both container-managed or bean-managed persistence, the JDBC Resource Adapter makes use of relational storage systems through the JDBC interface.

JDBC connections are obtained from a JDBC RA.

The JDBC RA implements the J2EE Connector Specification using the DataSource interface as defined in the JDBC [http://java.sun.com/javase/technologies/database/index.jsp] standard extensions.

An JDBC RA is configured to identify a database and a means to access it via a JDBC driver. Multiple JDBC RAs can be deployed either via the `jonas.properties` file or included in the `autoload` directory of the **resource** service.

The following section explains how JDBC RARs can be defined and configured in the JOnAS server.

To support distributed transactions, the JDBC RA requires the use of at least a JDBC2-XA-compliant driver. Such drivers implementing the XADataSource interface are not always available for all relational databases. The JDBC RA provides a generic driver-wrapper that emulates the XADataSource interface on a regular JDBC driver. It is important to note that this driver-wrapper does not ensure a real two-phase commit for distributed database transactions.

2.1.1. Generic JDBC Resource Adapters

The generic JDBC RAs of JOnAS provide implementations of the `java.sql.Driver`, `javax.sql.DataSource`, `javax.sql.ConnectionPoolDataSource`, and `javax.sql.XADataSource` interfaces. They are located in the `$JONAS_ROOT/rars/autoload` directory and thus are deployed automatically.They consist of base (or generic) RAs facilitating the build of the user JDBC RAs.

Depending on the relational database management server and the available interface in the used JDBC-compliant driver, the user JDBC RA is linked (through the RAR link feature) to a generic RA (for example, the Driver's one). In this case, the user RA contains only a jonas-ra.xml file with some specific parameters, such as the connection url, the user/password, or the JDBC-Driver class.

Resource adapter provided with JOnAS	description	jndi name
rars/autoload/ JOnAS_jdbcDS.rar	Generic JDBC RA that implements the `DataSource` interface	JOnASJDBC_DS
rars/autoload/ JOnAS_jdbcDM.rar	Generic JDBC RA that implements the `Driver` interface	JOnASJDBC_DM

rars/autoload/ JOnAS_jdbcCP.rar	Generic JDBC RA that implements the `ConnectionPoolDataSource` interface	JOnASJDBC_CP
rars/autoload/ JOnAS_jdbcXA.rar	Generic resource adapter that implements the `XADataSource` interface	JOnASJDBC_XA

2.1.2. Specific JDBC Resource Adapter

The remainder of this section, which describes how to define and configure JDBC RAs, is specific to JOnAS. However, the way to use these JDBC RAs in the Application Component methods is standard, i.e., via the resource manager connection factory references (refer to the example in the section Writing Database Access Operations [ejb2_programmer_guide.html#ejb2.bmp].

An RAR file must be deployed as explained in Section A.1, “resource service configuration”.

Usually a resource Adapter contains in its `rar` file all the classes needed to access to the external resource. In the case of a specific JDBC RA it contains only a JOnAS specific deployment descriptor `jonas-ra.xml` that tell what sort of generic resource adapter to use and information related to the specific database used. The `jar` file of the actual JDBC driver mut be copied in the right place to be seen by the JOnAS classloader : `$JONAS_BASE/lib/ext.`

Changing the configuration of the RA requires extracting and editing the deployment descriptor and updating the archive file. There are several possible ways to do this:

- With the RAConfig command (refer to the JOnAS Commands Reference Guide [command_guide.html] for a complete description of the command).

- Through the jonasAdmin console (refer to Administration guide for a complete description). In the jonasAdmin's tree, the Resource Adapter Module node (under the deployment node) contains a configure tab that allows editing of both the ra.xml file and the jonas-ra.xml file of the undeployed RA.

2.1.2.1. Defining the JOnAS Connector Deployment Descriptor: jonas-ra.xml

The `jonas-ra.xml` contains JOnAS specific information describing deployment information, logging, pooling, jdbc connections, and RAR config property values:

- *Deployment Tags*:

property name	description	possible values
jndiname	name the RAR will be registered as. This property is required. This value will be used in the resource-ref section of an Java EE composant.	• Anyname (for example jdbc_1)
rarlink	jndiname of a base RAR file. Useful for deploying multiple connection factories without having to deploy the complete RAR file again. When this is used, the only entry in RAR is a `META-INF/jonas-ra.xml`	• JONASJDBC_DM • JONASJDBC_DS • JONASJDBC_CP • JONASJDBC_XA

native-lib	defines the path where native libraries can be found.	• Any string for a path

- *Logging Tags*:

property name	description	possible values
log-enabled	determines if logging should be enabled for the RAR.	• False (default value) • True
log-topic:	defines the log topic that will be used to write log messages for this rar file.	• Any topic name • Default value is org.objectweb.jonas.jca

- *Pooling Tags*

property name	description	possible values
pool-init	Initial size of the managed connection pool	• 0 (default value) • n
pool-min	Minimum size of the managed connection pool.	• 0 (default value) • n
pool-max	Maximum size of the managed connection pool.	• n • -1 = unlimited (default value)
pool-max-age-minutes	Maximum number of minutes to keep the managed connection in the pool.	• 0 = an unlimited amount of time. • n in minutes
pstmt-max	Maximum number of PreparedStatements per managed connection in the pool. Only needed with the JDBC RA of JOnAS or another database vendor's RAR. Value of 0 is unlimited and -1 disables the cache.	• 0 = unlimited • n (default value = 10) • -1 = cache disabled
pstmt-cache-policy	Prepared statement cache policy.	• List (default value) : array based implementation of the cache • Map : map based implementation of the cache
pool-max-opentime	Identifies the maximum number of minutes that a managed connection can be left busy.	• 0 = an unlimited amount of time (default value). • n in minutes
pool-max-waiters:	identifies the maximum number of waiters for a managed connection. Default value is 0.	• 0 (default value) • n
pool-max-waittime	identifies the maximum number of seconds that a waiter will wait for a managed connection. Default value is 0.	• 0 (default value) • n in seconds

pool-sampling-period:	identifies the number of seconds that will occur between statistics samplings of the pool. Default is 30 seconds.	• n in seconds (default value = 30s)

- *JDBC Connection Tags*:

Note

Only valid for Connection implementation of java.sql.Connection.

property name	description	possible values
jdbc-check-level	Level of checking that will be done for the jdbc connection.	• 0 : no check (default value) • 1: check connection still open • 2 : send the test statement before reusing a connection from the pool • 3: (keep-alive feature) send the test statement on each connection every pool-sampling-period
jdbc-test-statement	Test SQL statement sent on the connection if the jdbc-check-level is greater than 1.	• A SQL statement

- *Config Property Value Tags:*

Each entry must correspond to the config-property specified in the ra.xml of the RAR file. The default values specified in the ra.xml will be loaded first and any values set in the jonas-ra.xml will override the specified defaults. These tags differs dependiing on the generic JDBC RA used

property name	description	possible values
dsClass	Name of the class implementing `java.sql.Driver`, `javax.sql.DataSource`, `javax.sql.ConnectionPoolDataSource`, or `javax.sql.XADataSource` interfaces in the JDBC driver.	• any classname representing a JDBC driver (example:org.postgresql.Driver)
URL	Database url of the form jdbc:<database_vendor_subprotocol>. This property may be used only for JDBC RA that implements the Driver (JDBC_DM)	• Any url valid for a database provider (example:jdbc:postgresql://localhost:5432/mydb)
user	Database user name	• any name
password:	Database password	• any string
loginTimeout	Maximum time in seconds that the driver will wait while attempting to connect to a database.	• no value = 0 (default value) • n in seconds
isolationLevel	Level of transaction isolation	• none

		• serializable • read_committed • read_uncommitted • repeatable_read
mapperName	Name of the JORM mapper	The possible values can be found in the List of available mappers in JORM documentation [http://jorm.objectweb.org/doc/mappers.html].
databaseName	Name of the database	• any name
description:	Informal description	• any String
portNumber	Port Number of the database server	• a number
serverName	Name of the database server.	• any name
dbSpecificMethods	allow flexibility to call setter methods on the dsClass as required by the database provider	see below the particular syntax

- dbSpecificMethods a specific property:

  The JOnAS JDBC Resource Adapter is built as a generic connector to any database provider. The limitation of this is that each database provider may have different requirements about the methods needed to configure the `dataSource` class. This dbSpecificMethods property was added to allow flexibility to call setter methods on the dsClass as required by the database provider. The specific information about what additional methods should be used is documented by the database provider. The format of the value specified is:

  [:<del_char>]<method>=<value>::<value_type>:<method>=<value>::<value_type>....with:

:	optional starting value that denotes using the next character as the delimiter instead of the default ':'
<del_char>	delimiter character to use
<method>	method to call followed by an = sign
<value>	the parameter value to pass to the method being called, followed by 2 delimiter characters.If a Properties object is being passed, then the format of this value must be (name=val, name=val, ...);
<value_type>	the parameter type used to construct the reflection call, followed by the delimiter character if additional methods are being called • Boolean or bool • Byte or byte • Character or char • Double or double

	• Float or float • Integer or int • Long or long • Properties or java.util.Properties • Short or short • String

Note

If this JDBC resource is used as a persistence unit, the persistence configuration defined in the `persistence.xml` file must be coherent to this `jonas-ra.xml` description, such as the datasource name and the dialect.

2.1.2.2. Understanding pooling tags:

At JDBC RA deployment time, if **pool-init** is not null **pool-init** JDBC connection are created.

When a user requests a jdbc connection, the JDBC RA first checks to see if a connection is already open for its transaction. If not, it tries to get a free connection from the free list. If there are no more connections available, it creates a new jdbc connection (if **pool-max** is not reached).

If it cannot create new connections, the user must wait (if **pool-max-waiters** is not reached) until a connection is released. After a limited time (**pool-max-waittime**), the `getConnection` returns an exception.

When the user calls `close()` on its connection, it is put back in the free list.

Many statistics are computed (every **pool-sampling-period** seconds) and can be viewed by JonasAdmin. This is useful for tuning these parameters and for seeing the server load at any time

When a connection has been open for a time too long (**pool-max-age**), the pool will try to release it from the freelist. However, the JDBC RA always tries to keep open at least the number of connections specified in **pool-min**.

When the user has forgotten to close a jdbc connection, the system can automatically close it, after **pool-max-opentime** minutes. Note that if the user tries to use this connection later, thinking it is still open, it will return an exception (socket closed).

When a connection is reused from the freelist, it is possible to verify that it is still valid. This is configured in **jdbc-check-level** . For levels >1 it tries a dummy statement on the connection before returning it to the caller. This statement is configured in **jdbc-test-statement**.

Note

this previous description is not only true for JDBC RAs but also for all types of resource adapters, except **jdbc-check-level** and **jdbc-test-statement** which are specifics for JDBC.

2.1.3. Examples of Specific JDBC Resource Adapter

2.1.3.1. Oracle JDBC resource adapter (Driver)

An RAR for Oracle named as `jdbc_1` in JNDI and using the Oracle thin Driver JDBC driver, should be described in a file (called `for example Oracle1_DM.rar`), with the following properties configured in the jonas-ra.xml file:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jonas-connector 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-connector_4_2.xsd" >
  <jndi-name>jdbc_1</jndi-name>
  <rarlink>JOnASJDBC_DM</rarlink>
  <jonas-config-property>
    <jonas-config-property-name>user</jonas-config-property-name>
    <jonas-config-property-value>scott</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>password</jonas-config-property-name>
    <jonas-config-property-value>tiger</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>loginTimeout</jonas-config-property-name>
    <jonas-config-property-value></jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>URL</jonas-config-property-name>
    <jonas-config-property-value>jdbc:oracle:thin:@malte:1521:ORA1</jonas-config-property-
value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>dsClass</jonas-config-property-name>
    <jonas-config-property-value>oracle.jdbc.driver.OracleDriver</jonas-config-property-
value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>mapperName</jonas-config-property-name>
    <jonas-config-property-value>rdb.oracle</jonas-config-property-value>
  </jonas-config-property>
</jonas-connector>
```

In this example, "malte" is the hostname of the server running the database Oracle, 1521 is the SQL*Net V2 port number on this server, and ORA1 is the ORACLE_SID. This example makes use of the Oracle "Thin" JDBC driver. For an application server running on the same host as the Oracle DBMS, you can use the Oracle OCI JDBC driver.

2.1.3.2. PostgreSQL JDBC resource adapter (Driver)

To create a PostgreSQL RAR configured as jdbc_3 in JNDI, it should be described in a file (called for examplePostgreSQL3_DM.rar), with the following properties configured in the jonas-ra.xml file:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jonas-connector 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-connector_4_2.xsd" >
  <jndi-name>jdbc_3</jndi-name>
  <rarlink>JOnASJDBC_DM</rarlink>
<jonas-config-property>
    <jonas-config-property-name>user</jonas-config-property-name>
    <jonas-config-property-value>jonas</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>password</jonas-config-property-name>
    <jonas-config-property-value>jonas</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>loginTimeout</jonas-config-property-name>
    <jonas-config-property-value></jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>URL</jonas-config-property-name>
    <jonas-config-property-value>jdbc:postgresql:/malte:5432/db_jonas</jonas-config-
property-value>
```

```
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>dsClass</jonas-config-property-name>
    <jonas-config-property-value>org.postgresql.Driver</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>mapperName</jonas-config-property-name>
    <jonas-config-property-value>rdb.postgres</jonas-config-property-value>
  </jonas-config-property>
</jonas-connector>
```

2.1.3.3. Oracle JDBC resource adapter (XADataSource)

An RAR for Oracle configured as jdbc_4 in JNDI and using the Oracle `XADataSource` interface of the JDBC driver thin in order to use a JDBC2-XA-compliant driver.It may be described in a file (called for example Oracle1_XA.rar), with the following properties configured in the jonas-ra.xml file:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jonas-connector 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-connector_4_2.xsd" >
  <jndi-name>jdbc_4</jndi-name>
  <rarlink>JOnASJDBC_XA</rarlink>
   <jonas-config-property>
      <jonas-config-property-name>user</jonas-config-property-name>
      <jonas-config-property-value>jonas</jonas-config-property-value>
   </jonas-config-property>
   <jonas-config-property>
      <jonas-config-property-name>password</jonas-config-property-name>
      <jonas-config-property-value>jonas</jonas-config-property-value>
   </jonas-config-property>
   <jonas-config-property>
      <jonas-config-property-name>databaseName</jonas-config-property-name>
      <jonas-config-property-value>dbjonas</jonas-config-property-value>
    </jonas-config-property>
    <jonas-config-property>
      <jonas-config-property-name>portNumber</jonas-config-property-name>
      <jonas-config-property-value>1521</jonas-config-property-value>
    </jonas-config-property>
    <jonas-config-property>
      <jonas-config-property-name>serverName</jonas-config-property-name>
      <jonas-config-property-value>wallis</jonas-config-property-value>
    </jonas-config-property>
    <jonas-config-property>
      <jonas-config-property-name>dbSpecificMethods</jonas-config-property-name>
      <jonas-config-property-value>:#setDriverType=thin##String</jonas-config-property-
value>
    </jonas-config-property>
    <jonas-config-property>
      <jonas-config-property-name>dsClass</jonas-config-property-name>
      <jonas-config-property-value>oracle.jdbc.xa.client.OracleXADataSource</jonas-config-
property-value>
    </jonas-config-property>
</jonas-connector>
```

2.1.4. Tracing SQL Requests through P6Spy

The P6Spy [http://www.p6spy.com/] tool provides an easy way to trace the SQL requests sent to the database.

To enable this tracing feature, perform the following configuration steps:

- Install the p6spy.jar[1] into $JONAS_BASE/lib/ext.
- Update the appropriate RAR file's `jonas-ra.xml` file by setting the dsClass property to `com.p6spy.engine.spy.P6SpyDriver`
- Set the realdriver property in the `spy.properties` file (located in `$JONAS_BASE/conf`) to the jdbc driver of your actual database.

- Verify that logger.org.objectweb.jonas.jdbc.sql.level is set to DEBUG in $JONAS_BASE/conf/trace.properties.

  Example jonas-ra.xml content:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jonas-connector 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-connector_4_2.xsd" >
  <jndi-name>jdbc_3</jndi-name>
  <rarlink>JOnASJDBC_DM</rarlink>
  <native-lib></native-lib>
  <log-enabled>true</log-enabled>
  <log-topic>org.objectweb.jonas.jdbc.DMPostgres</log-topic>
  <pool-params>
    <pool-init>0</pool-init>
    <pool-min>0</pool-min>
    <pool-max>100</pool-max>
    <pool-max-age>0</pool-max-age>
    <pstmt-max>10</pstmt-max>
  </pool-params>

 <jdbc-conn-params>
    <jdbc-check-level>0</jdbc-check-level>
    <jdbc-test-statement></jdbc-test-statement>
  </jdbc-conn-params>
  <jonas-config-property>
    <jonas-config-property-name>user</jonas-config-property-name>
    <jonas-config-property-value>jonas</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>password</jonas-config-property-name>
    <jonas-config-property-value>jonas</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>loginTimeout</jonas-config-property-name>
    <jonas-config-property-value></jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>URL</jonas-config-property-name>
    <jonas-config-property-value>jdbc:postgresql://your_host:port/your_db</jonas-config-
property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>dsClass</jonas-config-property-name>
    <jonas-config-property-value>com.p6spy.engine.spy.P6SpyDriver</jonas-config-property-
value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>mapperName</jonas-config-property-name>
    <jonas-config-property-value>rdb.postgres</jonas-config-property-value>
  </jonas-config-property>
  <jonas-config-property>
    <jonas-config-property-name>logTopic</jonas-config-property-name>
    <jonas-config-property-value>org.objectweb.jonas.jdbc.DMPostgres</jonas-config-
property-value>
  </jonas-config-property>
</jonas-connector>
```

  In `$JONAS_BASE/conf/spy.properties` file:

```
realdriver=org.postgresql.Driver
```

  In `$JONAS_BASE/conf/trace.properties`:

```
logger.org.objectweb.jonas.jdbc.sql.level  DEBUG
```

2.1.5. Migration from dbm service to the JDBC RA

The migration of a `Database.properties` file to a similar Resource Adapter can be accomplished through the execution of the following RAConfig tool command. Refer to the JOnAS Commands Reference Guide [command_guide.html#commands.raconfig] for a complete description of RAConfig command.

```
RAConfig -dm -p MySQL1 $JONAS_ROOT/rars/autoload/JOnAS_jdbcDM MySQL_dm
```

Generates a `MySQL_dm.rar` file linked to `JOnAS_jdbcDM.rar`, the `jonas-ra.xml` file inserted is created with values coming from the `ra.xml` file of the `JOnAS_jdbcDM.rar` and values from the `MySQL1.properties` file

The `jonas-ra.xml` created by the previous command can be updated further, if desired. Once the additional properties have been configured, update the `MySQL_dm.rar` file using the following command:

```
RAConfig -path .  MySQL_dm.rar 1

RAConfig -u jonas-ra.xml MySQL_dm.rar 2
```

1 Extraction of `jonas-ra.xml` of `MySQL_dm.rar` in the working directory
2 update `MySQL_dm.rar` with `jonas-ra.xml`

2.2. Configuring JMS Resource Adapters

JMS Resource adapters can be deployed, either via the JOnAS administration console, or via the `jonas admin -a` command, or by dropping the file in the `$JONAS_BASE/deploy` directory.

JMS connections are obtained from a JMS RA, which is configured to identify and access a JMS server.

The JORAM resource adapter archive (joram_ra_for_jonas-{joram.version}.rar) is provided with the JOnAS distribution. It is located in the `$JONAS_BASE/repositories/maven2-internal/org/objectweb/joram/joram_ra_for_jonas/{joram.version}/joram_ra_for_jonas-{joram.version}.rar` directory. This file has to be changed if a particular configuration is needed for JORAM.

By default, the joram.xml file, a deployment plan related to JORAM, is present in the `$JONAS_BASE/deploy` directory. This deployment plan is used to deploy JORAM. It declares among others the JORAM resource adapter archive to deploy.

2.2.1. JORAM Resource Adapter configuration files

The JORAM RA may be seen as the central authority to go through for connecting and using a JORAM platform. The RA is provided with a default deployment configuration which:

- Starts a collocated JORAM server in non-persistent mode, with id 0 and name s0, on host localhost and using port 16010; for doing so it relies on both an `a3server.xml` file located in the `$JONAS_BASE/conf` directory and the jonas-ra.xml file located within the RA.
- Creates managed JMS ConnectionFactory instances and binds them with the names **CF**, **QCF**, and **TCF**.
- Creates administered objects for this server (JMS destinations and non-managed factories) as described by the `joramAdmin.xml`, located in the `$JONAS_BASE/conf` directory; those objects are bound with the names **sampleQueue**, **sampleTopic**, **JCF**, **JQCF**, and **JTCF.**

The default configuration may, of course, be modified.

The JORAM integration into JOnAS is composed of 3 different parts: server, RA, and administration. Each part contains its own configuration files:

- a3servers.xml is the JORAM platform configuration file, i.e. the server part. The file is located in the `$JONAS_BASE/conf` directory.

- `ra.xml` and `jonas-ra.xml` are the resource adapter configuration files. They are embedded in the resource adapter (META-INF directory).
- `joramAdmin.xml` contains the administration tasks to be performed by the JORAM server such as the JMS objects creation. It is located in the `$JONAS_BASE/conf` directory.

2.2.1.1. JORAM server configuration : a3servers.xml

The a3server.xml (`$JONAS_BASE/conf/a3server.xml`) file describes the JORAM platform, i.e., the network domain, the used transport protocol, and the reachable JORAM servers. It is used by a JORAM server at start time. By default, only one collocated JORAM server is defined (s0) based on the tcp/ip protocol. A distributed configuration example is provided in the how-to document and other examples are available in JORAM's user guide.

```
<config>
  <property name="Transaction" value="fr.dyade.aaa.util.NullTransaction"/> 1
   <server id="0" name="S0" hostname="localhost"> 2
    <service class="org.objectweb.joram.mom.proxies.ConnectionManager"
             args="root root"/>
    <service class="org.objectweb.joram.mom.proxies.tcp.TcpProxyService"
             args="16010"/> 3

   </server>
</config>
```

1 This property means that the non persistent mode for JMS is choosen. In order to use persistent mode, the value must be changed to "fr.dyade.aaa.util.NTransaction"
2 Here can be set the server id and the host where the server run
3 args specifies the port number the JORAM server is listening on

The above configuration describes a JORAM platform made up of one unique JORAM server (id 0, name s0), running on localhost, listening on port 16010. Those values are taken into account by the JORAM server when starting. However, **they should match the values set in the deployment descriptor of the RA**, otherwise the adapter either will not connect to the JORAM server, or it will build improper connection factories.

The joram_raconfig command allows to modify these parameters in all the configuration files.

If used in non-collocated mode, joram can be started with the **JmsServer** command which loads the $JONAS_BASE/conf/a3server.xml configuration file.

2.2.1.2. Resource Adapter configuration: ra.xml, jonas-ra.xml

The `ra.xml` file is the standard deployment descriptor for the JORAM adapter and the `jonas-ra.xml` file is the JOnAS-specific deployment descriptor for the JORAM adapter. These files set the central configuration of the adapter, define and set managed connection factories for outbound communication, and define a listener for inbound communication. `jonas-ra.xml` contains specific parameters such as pool parameters or jndi names, but also may redefine the parameters of some `ra.xml` files and override their values. Globally, a good way to proceed is to keep the original `ra.xml` file with the default values and to customize the configuration only in the `jonas-ra.xml` file.

Changing the configuration of the RA requires extracting and editing the deployment descriptor and updating the archive file. There are several possible ways to do this:

- With the RAConfig command to extract jonas-ra.xml, do the following:

```
RAConfig -path . joram_for_jonas_ra.rar
```

  Then, to update the archive, do the following:

```
RAConfig -u jonas-ra.xml joram_for_jonas_ra.rar
```

- Through the jonasAdmin console (refer to Administration guide for a complete description).

  In the jonasAdmin's tree, the Resource Adapter Module node (under the deployment node) contains a configure tab that allows editing of both the ra.xml file and the jonas-ra.xml file of the undeployed RA.

- Through the joram_raconfig utility (refer to joram_raconfig description for a complete description).

  This tool allows easy modification to the network parameters of the JORAM server in all the configuration files.

The following properties are related to the central configuration of the adapter; they are set via some <jonas-config-property> elements:

property name	description	possible values
CollocatedServer	Running mode of the JORAM server to which the adapter gives access.	• True: when deploying, the adapter starts a collocated JORAM server. • False: when deploying, the adapter connects to a remote JORAM server. • Nothing (default True value is then set).
PlatformConfigDir	Directory where the `a3servers.xml` and `joramAdmin.xml` files are located.	• Any String describing an absolute path (ex: /myHome/myJonasRoot/conf). • Empty String, files will be searched in $JONAS_BASE/conf • Nothing (default empty string is then set).
PersistentPlatform	Persistence mode of the collocated JORAM server. - not taken into account if the JORAM server is set as non-collocated. - If true, set the property 'Transaction' to 'fr.dyade.aaa.util.NTransaction' before launching the JORAM server. - If false, set the property 'Transaction' to 'fr.dyade.aaa.util.NullTransaction' before launching the JORAM server. - Warning, if the 'Transaction' property is set in the a3server.xml file, this value is ignored.	• True: starts a persistent JORAM server. • False: starts a non-persistent JORAM server. • Nothing (default False value is then set).
ServerId	Identifier of the JORAM server to start (not taken into account if the JORAM server is set as non-collocated).	• Identifier corresponding to the server to start described in the a3servers.xml file (ex: 1). • Nothing (default 0 value is then set).

ServerName	Logical name of the JORAM server to start.In the collocated case, this parameter specifies the storage path of the persistent mode (absolute or relative path). If the JORAM server is non-collocated, it must be set to the name of the already started JORAM server (this is necessary for management purpose).	• Storage path of the persistent mode for the collocated case (ex: /tmp/s0). • Name of the started server as described in the a3servers.xml in the non collocated case (ex: s1) • Nothing (default s0 name is then set and the current directory is used for storing the persistent data).
AdminFileXML	Name of the file describing the administration tasks to be performed by the JORAM server, i.e., JMS destinations to create, users to create, ... If the file does not exist, or is not found, no administration task is performed.	• Name of the file (ex: myAdminFile.xml). • Nothing (default joramAdmin.xml name is then set).
HostName	Name of the host where the JORAM server runs, used for accessing a remote JORAM server (non-collocated mode), and for building appropriate connection factories.	• Any host name (ex: myHost). • Nothing (default localhost name is then set).
ServerPort	Port the JORAM server is listening on, used for accessing a remote JORAM server (non-collocated mode), and for building appropriate connection factories.	• Any port value (ex: 16030). • Nothing (default 16010 value is then set).
ConnectingTimer	Duration in seconds during which connecting is attempted (connecting might take time if the server is temporarily not reachable)	• 0 : set for connecting only once and aborting if connecting failed (default value) • n : duration in seconds
CnxPendingTimer	Period in milliseconds between two ping requests sent by the client connection to the server;	• 0 means "notimer" (default value) • n: duration in milliseconds
TxPendingTimer	Duration in seconds during which a JMS transacted (non XA) session might be pending; above that duration the session is rolled back and closed.	• 0 value means "no timer". • n: duration in seconds
DeleteDurableSubscription	Indicates the durable Subscriptions must be deleted when the consumer is closed	• True (previous behaviour) • False (default value)

The <jonas-connection-definition> elements wrap properties related to the managed connection factories:

There are three managed connection factories:

- A Queue managed connection factory registered in JNDI with the name **QCF**
- A Topic managed connection factory registered in JNDI with the name **TCF**
- A managed connection factory registered in JNDI with the name **CF**

Here are the properties that can be configured for each managed connection factory:

property name	description	possible values
jndi-name	Name used for binding the constructed connection factory.	Any name (ex: myQueueConnectionFactory). Default values are • **QCF** for the Queue managed connection factory • **TCF** for the Topic managed connection factory • **CF** for the managed connection factory
UserName	Default user name that will be used for opening JMS connections.	• Any name (ex: myName). • Nothing (default *anonymous* name will be set).
Password	Default user password that will be used for opening JMS connections.	• Any name (ex: myPass). • Nothing (default *anonymous* password will be set).
Collocated	Specifies if the connections that will be created from the factory should be TCP or local-optimized connections	• True (for building local-optimized connections). • False (for building TCP connections). • Nothing (default TCP mode will be set).

The <jonas-activationspec> element wraps a property related to inbound messaging:

property name	description	possible values
jndi-name	Binding name of a JORAM object to be used by 2.1 MDBs.	• Any name (by default:joramActivationSpec).

The Pooling Tags are the same than those for other RAs:

property name	description	possible values
pool-init	Initial size of the managed connection pool	• 0 (default value) • n
pool-min	Minimum size of the managed connection pool.	• 0 (default value) • n
pool-max	Maximum size of the managed connection pool.	• n • -1 = unlimited (default value)

pool-max-age-minutes	Maximum number of minutes to keep the managed connection in the pool.	• 0 = an unlimited amount of time. • n in minutes
pstmt-max	Maximum number of PreparedStatements per managed connection in the pool. Only needed with the JDBC RA of JOnAS or another database vendor's RAR. Value of 0 is unlimited and -1 disables the cache.	• 0 = unlimited • n (default value = 10) • -1 = cache disabled
pool-max-opentime	Identifies the maximum number of minutes that a managed connection can be left busy.	• 0 = an unlimited amount of time (default value). • n in minutes
pool-max-waiters:	identifies the maximum number of waiters for a managed connection. Default value is 0.	• 0 (default value) • n
pool-max-waittime	identifies the maximum number of seconds that a waiter will wait for a managed connection. Default value is 0.	• 0 (default value) • n in seconds
pool-sampling-period:	identifies the number of seconds that will occur between statistics samplings of the pool. Default is 30 seconds.	• n in seconds (default value = 30s)

2.2.1.3. JMS Applications Configuration

`joramAdmin.xml` file describes the configuration related to the application. It describes the administration objects in the JORAM server such as the JMS objects, the users, or the non-managed factories. In other words, it defines the JORAM objects to be (optionally) created when deploying the adapter.

In earlier version the `joram-admin.cfg` was used for this same purpose but it is now deprecated.

The default file provided with JOnAS creates a queue bound with the name *sampleQueue*, a topic bound with the name *sampleTopic*, sets the *anonymous* user, and creates and binds non-managed connection factories named *JCF*, *JQCF* and *JTCF*

Note

- All administration tasks are performed by the server connected but may affect remote JORAM servers to which the adapter is connected through the ServerId attribute.
- If a queue, a topic or a user already exists on the JORAM server (for example, because the server is in persistent mode and has re-started after a crash, or because the adapter has been deployed, undeployed and is re-deployed giving access to a remote JORAM server), it will be retrieved instead of being re-created.

The format of this file is XML. Here are some examples:

- simple example:

```
<?xml version="1.0"?>
```

```
      <JoramAdmin>
        <AdminModule>
          <collocatedConnect name="root" password="root"/>
        </AdminModule>
        <ConnectionFactory
className="org.objectweb.joram.client.jms.tcp.TcpConnectionFactory">
          <tcp host="localhost"
                  port="16010"/>
          <jndi name="JCF"/>
        </ConnectionFactory>
        <ConnectionFactory
className="org.objectweb.joram.client.jms.tcp.QueueTcpConnectionFactory">
          <tcp host="localhost"
                  port="16010"/>
          <jndi name="JQCF"/>
        </ConnectionFactory>
        <ConnectionFactory
className="org.objectweb.joram.client.jms.tcp.TopicTcpConnectionFactory">
          <tcp host="localhost"
                  port="16010"/>
          <jndi name="JTCF"/>
        </ConnectionFactory>
        <User name="anonymous"
                 password="anonymous"
                 serverId="0"/>
        <Queue name="sampleQueue">
          <freeReader/>
          <freeWriter/>
          <jndi name="sampleQueue"/>
        </Queue>
        <Topic name="sampleTopic">
          <freeReader/>
          <freeWriter/>
          <jndi name="sampleTopic"/>
        </Topic>
      </JoramAdmin>
```

- For requesting the creation of a new object, simply add the element in the file. For example, to add a queue 'MyQueue', add the following XML element:

```
        <Queue name="myQueue">
          <freeReader/>
          <freeWriter/>
          <jndi name="myQueue"/>
        </Queue>
```

- When the JORAM is not collocated, the AdminModule must be defined as follows:

```
      <AdminModule>
        <connect host="localhost"
                 port="16020"
                 name="root"
                 password="root"/>
      </AdminModule>
```

  The port number must be set with the server port number (defined in the `a3servers.xml` and in the JORAM's RAR configuration `ra.xml` and `jonas-ra.xml` files).

- Possible parameters for a queue definition:

```
       <Queue name=""
                 serverId=""
                 className=""
                 dmq=""
                 nbMaxMsg=""
                 threshold="">
          <property name="" value=""/>
          <property name="" value=""/>
          <reader user=""/>
          <writer user=""/>
          <freeReader/>
          <freeWriter/>
          <jndi name=""/>
          </Queue>
```

- Possible parameters for a topic definition:

```
<Topic name=""
          parent=""
          serverId=""
          className=""
          dmq="">
  <property name="" value=""/>
  <property name="" value=""/>
  <reader user=""/>
  <writer user=""/>
  <freeReader/>
  <freeWriter/>
  <jndi name=""/>
</Topic>
```

- Example of a dead message queue definition:

```
<DMQueue name="DMQ"
           serverId="0">
  <reader user="anonymous"/>
  <writer user="anonymous"/>
  <freeReader/>
  <freeWriter/>
  <jndi name="DMQ"/>
</DMQueue>
```

- Example of a scheduler queue definition:

```
<Destination type="queue"
               serverId="0"
               name="schedulerQueue"
               className="com.scalagent.joram.mom.dest.scheduler.SchedulerQueue">
  <freeReader/>
  <freeWriter/>
  <jndi name="schedulerQueue"/>
</Destination>
```

- Example of a clustered queues destination:

```
  <Cluster>
    <Queue name="queue0"
             serverId="0"
             className="org.objectweb.joram.mom.dest.ClusterQueue">
      <freeReader/>
      <freeWriter/>
      <property name="period" value="10000"/>
      <property name="producThreshold" value="50"/>
      <property name="consumThreshold" value="2"/>
      <property name="autoEvalThreshold" value="false"/>
      <property name="waitAfterClusterReq" value="1000"/>
      <jndi name="queue0"/>
    </Queue>
    <Queue name="queue1"
             serverId="1"
             className="org.objectweb.joram.mom.dest.ClusterQueue">
      <freeReader/>
      <freeWriter/>
      <property name="period" value="10000"/>
      <property name="producThreshold" value="50"/>
      <property name="consumThreshold" value="2"/>
      <property name="autoEvalThreshold" value="false"/>
      <property name="waitAfterClusterReq" value="1000"/>
      <jndi name="queue1"/>
    </Queue>
    <Queue name="queue2"
             serverId="2"
             className="org.objectweb.joram.mom.dest.ClusterQueue">
      <freeReader/>
      <freeWriter/>
      <property name="period" value="10000"/>
      <property name="producThreshold" value="50"/>
      <property name="consumThreshold" value="2"/>
      <property name="autoEvalThreshold" value="false"/>
      <property name="waitAfterClusterReq" value="1000"/>
      <jndi name="queue2"/>
    </Queue>
    <freeReader/>
    <freeWriter/>
```

```
        <reader user="user0"/>
        <writer user="user0"/>
        <reader user="user1"/>
        <writer user="user1"/>
        <reader user="user2"/>
        <writer user="user2"/>
      </Cluster>
```

2.2.1.4. joram_raconfig command

2.2.1.4.1. joram_raconfig

Change the host and port parameters of a given JORAM server in the configuration files.

2.2.1.4.1.1. Options

`joram_raconfig` [-p port] [-h host] [-s serverId]

-p port	Set the listening port of the JORAM server (defaults to 16010).
-h host	Set the IP address of the JORAM server (defaults to localhost).
-s serverId	Set the server id of the JORAM server (defaults to 0).

2.2.1.4.1.2. Description

The joram_raconfig tool aims to facilitate consistent updates (accross multiple files) for the host and port parameters of a given JORAM server ID.

JORAM relies on several configuration files: `a3servers.xml`, `joramAdmin.xml` and `ra.xml`. With joram_raconfig, these configuration files are updated all together and thus the consistency is ensured.

Modified files:

- `$JONAS_BASE/conf/a3servers.xml`
- `$JONAS_BASE/conf/joramAdmin.xml`
- `META-INF/ra.xml` (in the JORAM resource adapter) is updated.

Resource adapters files are looked up in the following places:

- `$JONAS_BASE/repositories/maven2-internal/org/objectweb/joram/joram_ra_for_jonas/{joram.version}/joram_ra_for_jonas-{joram.version}.rar`
- `$JONAS_BASE/deploy/joram_ra_for_jonas.rar`

2.2.1.4.1.3. Example

```
>$ joram_raconfig -h localhost -p 16012 -s 0
Target JORAM Resource Adapter: /home/ ... /joram/joram_ra_for_jonas/5.2.1a/
joram_ra_for_jonas-5.2.1a.rar
```

2.2.2. JORAM's Resource Adapter tuning

2.2.2.1. ManagedConnection Pool

A pool of ManagedConnection is defined for each factory (connection definition) specified in the jonas-ra.xml file. See the pool parameters in the Section 2.2.1.2, “Resource Adapter configuration: ra.xml, jonas-ra.xml” [19].

2.2.2.2. Session/Thread pool in the JORAM RA

The JORAM RA manages a pool of session/thread for each connection and, by default, the maximum number of parallel sessions is set to 10.

When linked with an message-driven bean, this maximum number of entries in the pool corresponds to the maximum number of messages that can be processed in parallel per message-driven bean. A session is released to the pool just after the message processing (onMessage()). When the maximum is reached, the inquiries for a session creation are blocked until a session becomes available in the pool.

The maxNumberOfWorks property can be set in the message-driven bean standard deployment descriptor. For example, the code below can be added to limit the number of parallel sessions to 100 (default value is 10).

```
<activation-config-property>
<activation-config-property-name>maxNumberOfWorks</activation-config-property-name>
<activation-config-property-value>100</activation-config-property-value>
</activation-config-property>
```

As this parameter set the max number of messages that can be treated simultaneously, the max-cache-size must be set accordingly in the specific deployment descriptor.

2.2.3. Undeploying and Redeploying a JORAM Adapter

Undeploying a JORAM adapter either stops the collocated JORAM server or disconnects from a remote JORAM server. It is then possible to deploy the same adapter again. If set for running a collocated server, it will re-start it. If the running mode is persistent, then the server will be retrieved in its pre-undeployment state (with the existing destinations, users, and possibly messages). If set for connecting to a remote server, the adapter will reconnect and access the destinations it previously created.

In the collocated persistent case, if the intent is to start a brand new JORAM server, its persistence directory should be removed. This directory is located in JOnAS' running directory and has the same name as the JORAM server (for example, s0/ for server "s0").

Appendix A. Appendix

A.1. resource service configuration

The **resource** service must be started when Java EE components require access to an external Enterprise Information Systems. The standard way to do this is to use a third party software component called **Resource Adapter**.

The role of the Resource service is to deploy the Resource Adapters in the JOnAS server, i.e., configure it in the operational environment and register in JNDI name space a connection factory instance that can be looked up by the application components. The **resource** service implements the Java EE Connector Architecture 1.5[1].

Resource Adapter are packaged in Java EE rar archives.

In development mode, as all other Java EE archives rar archives can be deployed automatically as soon as they are copied under $JONAS_BASE/deploy and undeployed as soon as they has been removed from this location.

For more information see ???.

The other ways to deploy rar archives is

- to use the jonasAdmin console.
- to use the command jonas admin:

```
jonas admin -a   <mydir>/myrar.rar
```

A JOnAS specific resource adapter configuration xml file must be included in each resource adapter. This file replicates the values of all configuration properties declared in the deployment descriptor for the resource adapter. Refer to Defining the JOnAS Connector Deployment Descriptor in J2EE Connector Programmer's Guide [connector_pg.html] for additional information.

Here is the part of `jonas.properties` related to **resource** service:

```
###################### JOnAS J2CA resource service configuration
#
#  Set the name of the implementation class of the J2CA resource service
jonas.service.resource.class org.ow2.jonas.resource.internal.JOnASResourceService
```

The worker thread pool used for all J2CA 1.5 Resource Adapters deployed can be configured in the ??? service.

resource service is mainly used in JOnAS for accessing databases via a JDBC resource adapter (in this case it replace **dbm** service) and for providing JMS facilities.

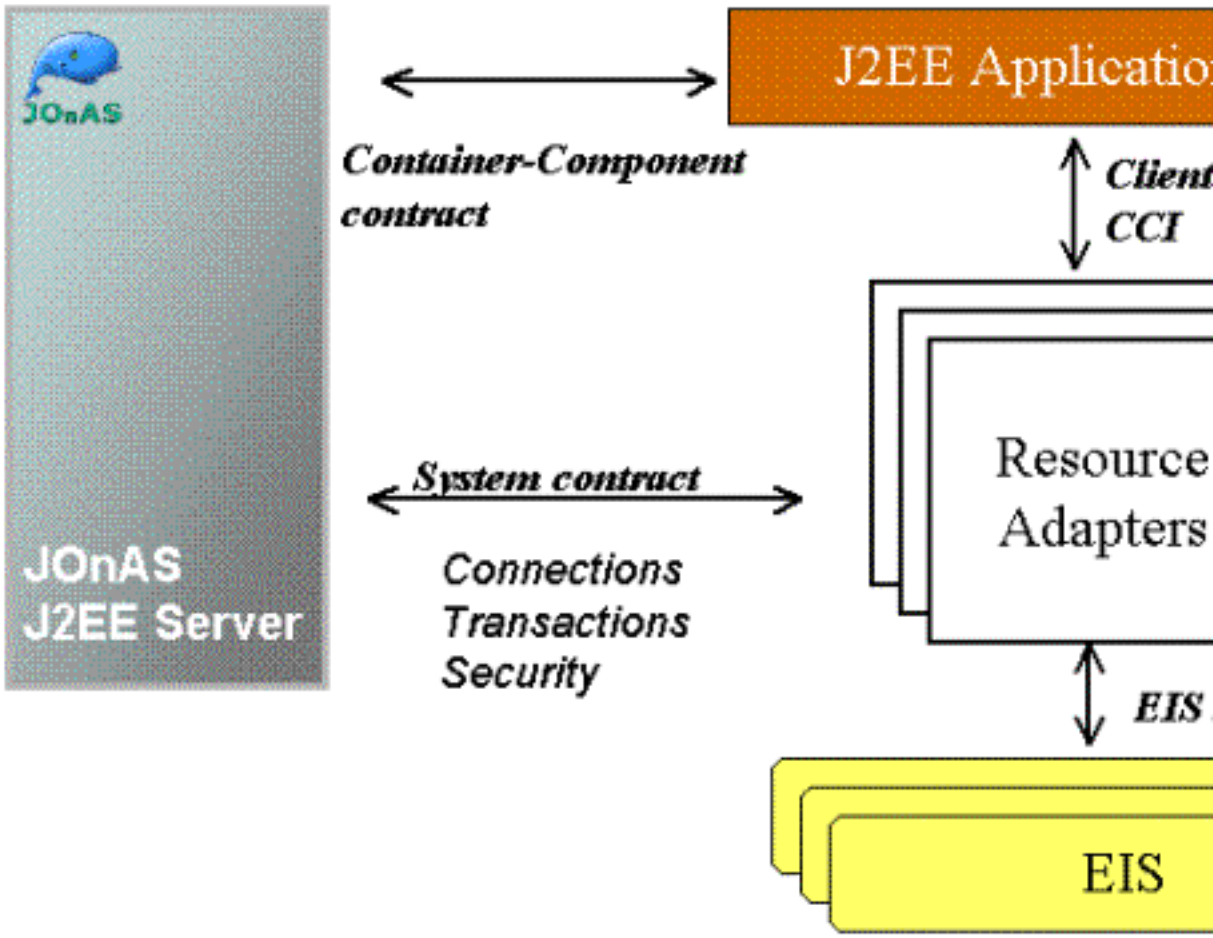
JOnAS provides several JDBC resource adapters and a JMS resource adapter on top of JORAM [http://joram.objectweb.org/] More information about configuring resource adapters can be found in Section 2.1, “Configuring JDBC Resource Adapters”

A.2. Connector Architecture Principles

The Java Connector Architecture allows the connection of different Enterprise Information Systems (EIS) to an application server such as JOnAS. It defines a way for enterprise applications (based on EJB, servlet, JSP or J2EE clients) to communicate with existing EIS. This requires the use of a third party software component called "Resource Adapter" for each type of EIS, which should be previously

[1]There is no real acronym for this specification JCA was the acronym for Java Cryptography Architecture . In the rest of this document we will use J2CA

deployed on the application server. The Resource Adapter is an architecture component comparable to a software driver, which connects the EIS, the application server, and the enterprise application (J2EE components in the case of JOnAS as application server). The RA provides an interface (the Common Client Interface or CCI) to the enterprise application (J2EE components) for accessing the EIS. The RA also provides standard interfaces for plugging into the application server, so that they can collaborate to keep all system-level mechanisms (transactions, security, and connection management) transparent from the application components.



The resource adapter plugs into JOnAS and provides connectivity between the EIS, JOnAS, and the application. The application performs "business logic" operations on the EIS data using the RA client API (CCI), while transactions, connections (including pooling), and security on the EIS is managed by JOnAS through the RA (system contract).