

# Installation

## 1. Overview

RADRunner is a pure J2EE application. Hence it is supplied as an EAR file (*radrunner.ear*), which can be extracted from the installation archive and deployed on any J2EE-compliant application server.

RADRunner has been tested with:

- Microsoft Windows (Windows 98 upwards), IBM AIX (versions 5.2 upwards), and SuSE Linux (versions 8.2 upwards) operating systems
- JBoss (we recommend [version 3.0.8](#)) and IBM WebSphere (versions 4.0.3 upwards) application servers
- MySQL (4.1a upwards) and IBM DB2 (7.2 upwards) relational databases
- Sun's Jini Software Kit (1.2.1) and GigaSpaces' JSpaces (2.1b) Jini implementations (*as a storage alternative to a relational database, a Jini JavaSpace can be used*)

To evaluate the product, we recommend [installing RADRunner with JBoss and MySQL on Windows](#). This is free of licence charges for third-party software.

For a configuration proven (by IBM Solution Partnership Centre, Hursley, UK) to be robust and scalable to 250,000 transactions an hour on low-cost hardware, and above as required, we recommend [installing RADRunner with IBM WebSphere and DB2 on Windows](#). Many other configurations will support equivalent loads - please contact us for advice.

Step-by-step instructions for each of the recommended installations are given below.

## 2. Quick Start

### 2.1. Installing RADRunner with JBoss and MySQL on Windows

With the free open source application server JBoss and the free open source database MySQL on Windows, you can install RADRunner as follows:

*Some versions (3.2.4 and 3.2.5) of JBoss have a bug that prevents RADRunner from executing (technical details). JBoss versions 3.2.3 and earlier are OK, but those are longer available on the JBoss web site. Hence we recommend that you return to an earlier version of*

JBoss, 3.0.8, which we currently use for production installations, or a version higher than 3.2.6RC2.

1. Download JBoss, and unzip to install (typically into C:\JBoss)
2. Download MySQL, and double-click to install (typically into C:\mysql)  
*Make sure to choose a Max version, since RADRunner uses database transactions*
3. Download a recent JDBC driver for MySQL, and extract from the zip the file *mysql-connector-java-\*-bin.jar* into C:\JBoss\server\default\lib  
*Make sure that the file is placed directly in this folder, and not into a sub-folder created by the unzip utility*
4. Double-click on *install\_radrunner.bat* to create a database and install RADRunner to JBoss  
*This script carries out the following actions:*
  - Create a MySQL database called *radrunner* with no username and password
  - Install a MySQL data source in JBoss
  - Deploy RADRunner to JBoss

To start JBoss, and hence the RADRunner application, double-click on *radrunner.bat*. Once JBoss has fully initialized, a message of the form "Started in Nm:NNs:NNNms" will appear in the JBoss console window. You will then be able to access the RADRunner login page by entering into a browser the following URL:

*http://localhost:8080/radrunner/RMLogin.jsp*

To stop JBoss, close the JBoss console window.

To avoid the need for a JBoss console window, you can install JBoss as a Windows Service. First set a Windows environment variable *JBOSS\_DIST* to your JBoss installation folder via *Start/Settings/Control Panel/System/Advanced/Environment Variables*. Then download one of the many free service creation tools available (for example, ServiceInstaller), and use it to create a service with the following settings, replacing the variables enclosed in % marks with the appropriate environment variable values on your system:

<i>Program</i>	%JAVA_HOME%\bin\java.exe
<i>Arguments</i>	"-Djava.security.policy=%JBOSS_DIST%\server\default\conf\serve "-Dcom.rolemodellers.jdbc.useInternalConnectionPool=false" -Dprogram.name=run.bat -classpath "%JAVA_HOME%\lib\tools.jar;%JBOSS_DIST%\bin\run.jar" org.jboss.Main
<i>Working Directory</i>	%JBOSS_DIST%\bin

## 2.2. Installing RADRunner with IBM WebSphere and DB2 on Windows

## Installation

With IBM WebSphere and DB2 on Windows, you can install RADRunner as follows:

1. Use DB2 Control Center to create a database in DB2, noting a user/password combination for it
2. Use the WebSphere Administrator's Console to create an application server with the following Java Virtual Machine (JVM) system properties (replace all text in <> with your own text):
  - `com.rolemodellers.jdbc.url=jdbc:db2://<server name>/<DB2 database name>`
  - `com.rolemodellers.jdbc.driver=COM.ibm.db2.jdbc.net.DB2Driver`
  - `com.rolemodellers.jdbc.username=<DB2 database username>`
  - `com.rolemodellers.jdbc.password=<DB2 database password>`
  - `com.rolemodellers.jdbc.useClob=true`
  - `com.rolemodellers.jdbc.maxPlaywrightLength=1000000`
  - `com.rolemodellers.jdbc.useCreateTableAppendText=false`
  - `com.rolemodellers.rim.mappingModel=http://localhost/radrunner/castor/mappingModel.xml`
  - `javax.naming.Context.INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory`
  - `java.naming.provider.url=iiop://<server name>:900`
  - `java.security.policy=<RADRunner installation package>\SecurityPolicy\rm.policy`
3. Extract `radrunner.ear`, and use the WebSphere Administrator's Console to install it to this application server
  - *When prompted for a Data Source, use any - SampleDataSource, for instance. Nothing will be written there.*
  - *There is no need to re-generate deployment code - you can specify either Yes or No when prompted.*

You will then be able to access the RADRunner login page by entering into a browser the following URL:

`http://localhost/radrunner/RMLogin.jsp`

*Note that this configuration is using DB2 to store application data. Your IBM licence may or may not cover this usage of DB2, so it is necessary to check before installing RADRunner in a production environment with the above configuration. You may prefer to use a [different database](#).*

### 2.3. Other Configurations

if you are running a different operating system from Windows, the only change required to the above configurations is to set the [Application Directories](#) appropriately (the default values are Windows-specific).

If you wish to use a different application server from those above, or have other requirements (such as to use an HTTP proxy server, or to configure Java 2 security more precisely), see the

detailed instructions given below which cover the more general case.

We have tried to make the RADRunner installation instructions as helpful and comprehensive as possible. Any [feedback](#) will be welcomed - either positive or negative. In particular, we would be interested to hear of people's experiences with deploying RADRunner on other application servers than the ones above.

## 3. Database

In its default configuration, RADRunner persists data via a relational database. This is the recommended persistence configuration for most installations, and it is assumed as the default throughout this document. However, it is also possible to set up RADRunner to [use Jini to persist data](#), which removes the need for a relational database entirely - if you decide to do this, skip the rest of this section and see [JINI](#).

### 3.1. Choosing a Database

RADRunner requires a database which supports:

- Some form of [long text](#) data type
- [Transactions](#).

#### 3.1.1. Long Text Support

RADRunner is able to use the standard SQL CLOB data type (*Character Large Object*) to store XML strings. To make use of this, a Java program must be able to read and write CLOBs to the database. This facility is not available in all databases.

For example, DB2 and Oracle both support CLOBs via Java, but with regard to *open source databases*, at the time of writing:

- The only open source database to provide CLOB support via Java is MySQL 4.1.0 Alpha, using the latest JDBC driver MySQL Connector/J 3.1.0 Alpha, although this is not yet a stable product.
- The Firebird database does provide CLOB support, but its JDBC driver JayBird does not.
- Postgres does not provide CLOB support in accordance with the SQL-99 specification.

Hence, if you wish to use one of the above databases, or another database that does not provide CLOB support via Java, it is necessary to set certain system properties. These settings instruct RADRunner not to use the CLOB data type, but another suitable type which is supported by the database you intend to use. For example, with MySQL you could set:

1. `com.rolemodellers.jdbc.useClob=false`
2. `com.rolemodellers.jdbc.playwrightColumnType=LONGTEXT`

## Installation

### 3. *com.rolemodellers.jdbc.maxPlaywrightLength=0*

Setting *maxPlaywrightLength* to zero ensures that the column will be defined simply as of type LONGTEXT, rather than erroneously as of type LONGTEXT(length).

#### 3.1.2. Transaction Support

RADRunner does not save changes to the database that are made in a task, or series of tasks, until the enclosing activity is saved. To permit this, the underlying database must itself support transactions.

Most databases provide transaction support. However, with some databases it is an optional feature.

In particular, MySQL in its default configuration does not provide transaction support. To enable it, you must use a *Max* version of MySQL, and create the RADRunner table with *TYPE = InnoDB*. This is done automatically by the setup script for the default evaluation configuration, [Installing RADRunner with JBoss and MySQL on Windows](#) - see [Configuring the database](#) for details.

## 3.2. Using the database from RADRunner

RADRunner provides 2 options for using a relational database. Both of these require some system properties to be specified - see [System Properties](#) for details of how to do this.

Firstly, you can specify that RADRunner should bypass the application server and access the database directly. This is done via the system properties:

1. *com.rolemodellers.jdbc.useInternalConnectionPool=true*
2. *com.rolemodellers.jdbc.driver=databaseDriverClassName*
3. *com.rolemodellers.jdbc.url=databaseUrl*
4. *com.rolemodellers.jdbc.username=databaseUsername*
5. *com.rolemodellers.jdbc.password=databasePassword*

In this case, RADRunner implements its own internal connection pooling and transaction management, and any database driver and application server can be used. There is no performance or security penalty to using this approach, but it means that the standard J2EE configuration mechanisms for the data source are not available via the application server. RADRunner can still be used to configure some basic parameters, via the system properties:

1. *com.rolemodellers.jdbc.connectionTimeout=timeoutInMilliseconds*
2. *com.rolemodellers.jdbc.connectionPoolInitialSize=initialSize*
3. *com.rolemodellers.jdbc.connectionPoolReapInterval=reapConnectionsEverySoManyMilliseconds*
4. *com.rolemodellers.jdbc.transactionManagerReapInterval=reapTransactionsEverySoManyMilliseconds*

If this approach is used, configure settings via the relational database management system itself, not the application server. With DB2, for instance, it may be necessary to increase both the *maximum number of agents* on the instance and *maximum number of applications* on the database, depending on how heavy the usage is from RADRunner.

This approach is the default, and is recommended with WebSphere.

Secondly, a Data Source can be set up in your application server, and its JNDI name referenced from RADRunner. This is only recommended with application servers where the Transaction Manager can be accessed via JNDI (such as, for example, JBoss and WebLogic). The approach will also work with WebSphere, for example, but is not recommended since the implementation is necessarily low-level and may not be portable across different versions of the application server.

However, RADRunner creates transactions which span multiple servlet requests, and this is only possible if:

1. Your database driver supports JDBC 2. The recent releases of most major databases do this.  
and
2. The transaction manager implementation in your application server allows a suspended transaction to be resumed by a different thread. This is not mandated by the Java Transaction API specification - however, it is supported by JBoss and some versions of WebSphere, for example.

*If you are not sure about either of these with your database and application server, please [contact us for support](#).*

All that is required in RADRunner to set up this first option is to assign the system property *com.rolemodellers.jdbc.dataSource* to the JNDI name of your Data Source - see [System Properties](#) for details of how to do this.

The means of setting up a Data Source is specific to your application server. With JBoss, an appropriately customised XML file should be placed in the *deploy* directory. With WebSphere, it can be done from the administration console.

### 3.3. Configuring the database

However RADRunner accesses the database, it uses a single table to store all data. The default format of this table is:

```
CREATE TABLE RM_PLAYWRIGHT_OBJECT (  
  RMPO_WORLD VARCHAR(100) NOT NULL,  
  RMPO_ID VARCHAR(41) NOT NULL,
```

## Installation

```
RMPO_NAME VARCHAR(100) NOT NULL,  
RMPO_CLASS_ID VARCHAR(41),  
RMPO_CLASS_NAME VARCHAR(100),  
RMPO_SUPERCLASS_ID VARCHAR(41),  
RMPO_SUPERCLASS_NAME VARCHAR(100),  
RMPO_EMAIL VARCHAR(100),  
RMPO_PASSWORD VARCHAR(100),  
RMPO_TYPE VARCHAR(100) NOT NULL,  
RMPO_PLAYWRIGHT CLOB(100000) NOT NULL,  
RMPO_CREATED TIMESTAMP NOT NULL,  
RMPO_LAST_UPDATED TIMESTAMP NOT NULL  
)
```

The name of this table, the column prefix, names of all the columns, and the lengths of most columns can be specified using the system properties:

```
com.rolemodellers.jdbc.table  
com.rolemodellers.jdbc.columnPrefix  
com.rolemodellers.jdbc.worldColumn  
com.rolemodellers.jdbc.idColumn  
com.rolemodellers.jdbc.nameColumn  
com.rolemodellers.jdbc.classIdColumn  
com.rolemodellers.jdbc.classNameColumn  
com.rolemodellers.jdbc.superclassIdColumn  
com.rolemodellers.jdbc.superclassNameColumn  
com.rolemodellers.jdbc.emailColumn  
com.rolemodellers.jdbc.passwordColumn  
com.rolemodellers.jdbc.typeColumn  
com.rolemodellers.jdbc.playwrightColumn  
com.rolemodellers.jdbc.createdColumn  
com.rolemodellers.jdbc.lastUpdatedColumn  
com.rolemodellers.jdbc.maxWorldLength  
com.rolemodellers.jdbc.maxNameLength  
com.rolemodellers.jdbc.maxEmailLength  
com.rolemodellers.jdbc.maxPasswordLength  
com.rolemodellers.jdbc.maxPlaywrightLength
```

It is also possible to control database-specific aspects of the table creation via the system properties:

```
com.rolemodellers.jdbc.useCreateTableAppendText  
com.rolemodellers.jdbc.createTableAppendText
```

If *com.rolemodellers.jdbc.useCreateTableAppendText* is set to *true*, the value of *com.rolemodellers.jdbc.createTableAppendText* is appended to the SQL *CREATE TABLE* statement. For example, with MySQL *com.rolemodellers.jdbc.createTableAppendText* can be set to "Type = InnoDB" in order to enable transaction support (with the Max version of MySQL). You can also use this setting to specify such details as the tablespace to be used, disk quotas, block size, and so on.

Alternatively, you can create the table yourself outside of RADRunner, in order to exercise

full control over its properties - file placement, for example. If you create the table itself, it is advisable to create:

- a unique index on the id column
- a non-unique index on the email and password columns

Otherwise RADRunner will create the table and indexes the first time it starts up, using the defaults assigned to the schema with which it accesses the database.

See [System Properties](#) for details of how to set up all the above system properties, and [I/O Bottlenecks](#) for some guidelines on setting up a database for RADRunner.

## 4. Jini

As an alternative to using a relational database, RADRunner is able to persist data via a Jini JavaSpace, using a Jini Transaction Manager. This feature has been developed using Sun's freely available *Jini Software Kit* (JSK), version 1.2.1, and tested on both the JSK and the commercial product *JSpaces* available from GigaSpaces.

A Jini persistence configuration may be valuable in installations which require the data store to run on low-specification hardware, since Jini was originally developed as a solution appropriate for embedded devices. The free JSK in particular is optimized to run very fast on small systems.

If, however, you wish to use a Jini persistence solution with a high-throughput installation such as an application server cluster, it will be necessary to use JSpaces rather than the free JSK. JSpaces is a commercial product which was designed from the ground up to scale, whereas there is a known problem with Sun's Jini Software Kit (JSK) in versions up to 1.2.1, which can cause the JavaSpace and related services to fail in situations of exceptionally high and sustained load. Sun say that this problem will be rectified in future releases of the JSK, but at present our testing demonstrates the JSK version 1.2.1 to be unstable when loads of over 1500 requests/minute are sustained for several minutes in a row.

*Please note that RADRunner with Jini will not work with versions of WebSphere prior to 4.0.3, as support for Java 2 Security was only introduced in this version. It should be possible to install RADRunner with the database persistence option on any application server.*

We assume in the rest of this section that you wish to install RADRunner using the JSK. If you wish:

*Firstly*, download the latest version of the JSK, and unpack it on a suitable server machine. Some example Windows batch files are supplied in case you wish to use them to run Jini services. Before doing so, carry out the following 3 steps.

## Installation

With *JSpaces*, download the latest version of the software from the GigaSpaces web site.

Secondly, it is necessary to make certain JAR files supplied in the JSK (those of the form *\*-dl.jar* in directory *lib*) available via HTTP on your network, so that RADRunner - which is a client via RMI of the Jini services - can download them when necessary. This can be done (for example) by making a sub-directory *jini* under the document root of a web server on your network, then copying those JAR files to this sub-directory. Alternatively, it may be possible to configure a web server to serve the JARs directly from the Jini Software Kit *lib* directory.

With *JSpaces*, use the downloadable jars supplied by GigaSpaces, and make them available on your HTTP server under directory *lib* (e.g., at URL *http://localhost:8080/lib*). It is also necessary to add the GigaSpaces file *JSpaces-dl.jar* to the EAR file (without a path).

Thirdly, set the following operating system environment variables:

Make sure that these variables are capitalized correctly - some are used in security policy validation, which is case sensitive. For instance, *c:\myJavaHome* is NOT the same as *C:\myJavaHome*.

Fourthly, create a directory called *log* under *RM\_JINI\_HOME*.. Make sure that neither *RM\_JINI\_HOME\log* nor any of its parent directories are *read-only*, or the services will not be able to write their log files and will fail to run.

The batch files supplied are:

Once *RemoveWorldAndInitializeJini.bat* has been run an initial time, do not run it again unless you wish to delete all data in the JavaSpace and start from scratch. To start the services up in a normal way, just run *StartJini.bat* - this will restart the services from the log files.

*Neither the third nor the fourth step is necessary with JSpaces. Instead, start the GigaSpaces server, and create an appropriately named JavaSpace by hand.*

Fifth and finally, in order that RADRunner can access the Jini services as a client, it is necessary to set various system properties in the application server Java Virtual Machine (see [System Properties](#) for how to do this):

1. *com.rolemodellers.jini.codeBase=jiniCodebase*. This is the codeBase used to serve the downloadable JARs for RMI, and should be the same value that is specified by the operating system environment variable *RM\_JINI\_CODEBASE*.
2. *com.rolemodellers.jini.codeHost=jiniJarServer*. This is the name or IP address of the machine used to serve the downloadable JARs for RMI, and forms the first part of the codeBase.

3. *com.rolemodellers.jini.worldHost=jiniServer*. This is the name or IP address of the machine used to host the Jini services, and may or may not be the same as the codeHost.
4. *com.rolemodellers.rim.worldName=javaSpaceName*. This is the name of your Playwright World.

*This fifth step is necessary both with the JSK and with JSpaces.*

#### **4.1. Using JSpaces Instead Of The JSK**

*Download the software* from the GigaSpaces web site instead of from Sun.

JSpaces requires by default that the following *downloadable jars* (which it supplies) will be available on your HTTP server under directory *lib* (e.g., at URL *http://localhost:8080/lib*):

It is also necessary to add the GigaSpaces file *JSpaces-dl.jar* to the EAR file (without a path).

*This replaces the need to make the JSK downloadable jars available via HTTP.*

It is necessary to *explicitly create a JavaSpace* before starting RADRunner, using the supplied utility *createSpace*. The name of the space should then be assigned to the RADRunner system property *com.rolemodellers.rim.worldName* - the default assumed by RADRunner is *RoleModellers*.

*This is not necessary with the JSK, which creates a JavaSpace of the name that you specify when the services are initialized.*

It is necessary to *start the GigaSpaces server* before starting RADRunner. This can be done either by executing a command line, or (on Windows) by setting up the GigaSpaces server to run as an operating system service.

*With the JSK, the RMI daemon must be started from the command line to run the services.*

### **5. Application Directories**

RADRunner uses various directories on the server to read and write data. By default, running on Windows 2000, the following folders are used:

You may wish to use differently named or structured directories. If they do not exist when RADRunner starts up, it will create them. If they do already exist, make sure that none of the directories are *read-only*, or RADRunner will not be able to write to the filesystem and will fail to run.

Once you have chosen suitable directories on your network, it is necessary to set various system properties in the application server Java Virtual Machine (see [System Properties](#) for

## Installation

how to do this):

- *com.rolemodellers.rim.logPath*
- *com.rolemodellers.rim.sessionPath*
- *com.rolemodellers.rim.outPath*
- *com.rolemodellers.rim.inPath*

Each value set above should ideally include the full network path, it is necessary to escape backslashes, and the *inPath* should be accessible via HTTP if you wish to uploaded files to be visible from within a role instance.

For example, on a Windows system running Apache web server you might set:

```
com.rolemodellers.rim.logPath=\\\\myServer\C$\Program          Files\Apache
Group\Apache\htdocs\rim\log\
com.rolemodellers.rim.sessionPath=\\\\myServer\C$\Program    Files\Apache
Group\Apache\htdocs\rim\session\
com.rolemodellers.rim.outPath=\\\\myServer\C$\Program        Files\Apache
Group\Apache\htdocs\rim\out\
com.rolemodellers.rim.inPath=\\\\myServer\C$\Program         Files\Apache
Group\Apache\htdocs\rim\in\
```

If you do not wish to install as complex a web server as Apache, there are alternative freeware products available. For example, on Windows there is the very small and simple application TinyWeb.

To make these filesystem locations visible via HTTP, you must set further system properties, to specify the text prefixed to each filename in order to generate a URL to that file. For example, you might set:

```
com.rolemodellers.rim.logUrlPrefix=http://myHostName:myPort/rim/log/
com.rolemodellers.rim.sessionUrlPrefix=http://myHostName:myPort/rim/session/
com.rolemodellers.rim.outUrlPrefix=http://myHostName:myPort/rim/out/
com.rolemodellers.rim.inUrlPrefix=http://myHostName:myPort/rim/in/
```

In order to enable uploaded files to be visible from within a role instance outside the same LAN as the server, you must:

1. Set *inUrlPrefix*
2. Configure a web server to make the *in* directory visible via this prefix.

If you wish to make Role and resource snapshots visible outside the LAN, you must:

1. Set *sessionUrlPrefix*
2. Configure a web server to make the *session* directory visible via this prefix.
3. Set system property *com.rolemodellers.rim.snapshotProtocol=http*.

The *in* directory is used to store files uploaded to the server by Role instances. It is the system administrator's responsibility periodically to clear from this directory files that have become obsolete - for example, those to which no current Role instance refers. This can be detected via a text search on the world in exported format.

## 6. Application Server Configuration

### 6.1. Java 2 Security

RADRunner loads various classes dynamically, so it is necessary to enable Java 2 Security. You must specify a security policy file somewhere on your network. This could be based on an example file supplied with the application server - for example, WebSphere supplies the file *WebSphereInstallationRoot\properties\java.policy*. Alternatively, a sample security policy file *rm.policy* which could be used with either JBoss or WebSphere is included in the RADRunner installation package, under directory *SecurityPolicy*.

If you are using Jini, it is necessary to ensure that the *security policy file* for the application server grants the necessary permissions to the Jini codeBase (on top of any default permissions that should be granted for that application server). Then include in the file the following additional lines:

```
grant codeBase "${com.rolemodellers.jini.codeBase}/-" {
    permission java.security.AllPermission;
};

grant {
    permission java.net.SocketPermission "${com.rolemodellers.jini.codeHost}:*",
"connect,accept,resolve";

    permission java.net.SocketPermission "127.0.0.1:*",
"connect,accept,resolve";

    permission java.io.FilePermission "${com.rolemodellers.rim.logPath}/-",
"read,write,delete";

    permission java.io.FilePermission "${com.rolemodellers.rim.sessionPath}/-",
"read,write,delete";

    permission java.io.FilePermission "${com.rolemodellers.rim.outPath}/-",
"read,write,delete";

    permission java.io.FilePermission "${com.rolemodellers.rim.inPath}/-",
```

```
"read,write,delete";  
};
```

### 6.2. System Properties

Various *system properties* must be set in the JVM of the application server on which RADRunner is deployed. This can be done in one of 2 ways:

1. In the application server itself, by adding properties of the form `-DpropertyName=propertyValue` to the command line. It may also be possible to specify system properties from a dialog within the administration console (as with WebSphere). If this approach is taken, the application server configuration should be exported and backed up for future use - for example, with WebSphere this can be done via the tool *XMLConfig*.
2. By editing the EAR file. The EAR contains a file *radrunner-ejb.jar*, which itself contains a properties file, *com/rolemodellers/openapi/RMRimProperties.properties*. This approach would allow the updated EAR to be deployed on multiple application servers in the same environment with minimum additional effort.

If you wish to use the second approach, it will be necessary to edit *radrunner.ear*. This can be done with any common zip file tool - on Windows, you could use WinZip, for example. Open *radrunner.ear*, then open the file *radrunner-ejb.jar* within it. Extract *RMRimProperties.properties* into a new, empty directory on your hard disk - a sub-directory *com\rolemodellers\openapi* will be created by the extraction process, and the file placed in it. Leaving *radrunner-ejb.jar* open, edit the file to set properties as required. Then add *the original directory you created* to *radrunner-ejb.jar* - with WinZip you would use the command "Add with wildcards" with "Include subfolders" checked, and specify a filename of `*.*`. This will ensure that the properties file has the correct path inside the jar. Close *radrunner-ejb.jar*, choosing "Yes" when asked if you wish to update *radrunner.ear*, then close *radrunner.ear*.

The properties that *must* be set are all those specified in [DATABASE](#) (if you are using a database persistence option), [JINI](#) (if you are using a Jini persistence option) and [APPLICATION DIRECTORIES](#), plus:

1. *proxySet*, *proxyHost* and *proxyPort*:  
If no HTTP proxy is used on your network, make *proxySet=false*, and do not set *proxyHost* and *proxyPort*.  
If an HTTP proxy is used on your network, make *proxySet=true* and set *proxyHost* and *proxyPort* appropriately.
2. The *JNDI properties* for the application server you wish to use - for example: (with WebSphere)  
`javax.naming.Context.INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory`  
`java.naming.provider.url=iiop://<server name>:900`

(with JBoss)

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
```

```
java.naming.provider.url=jnp://localhost:1099
```

```
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
```

If these are not set appropriately for your application server, RADRunner will not be able to look up references via JNDI (for example, to EJBs or Data Sources), and you will not be able to log in. Make sure that any JNDI properties not appropriate to your application server are deleted or commented out in the EAR file.

Other properties that *may* need to be set are:

1. If you wish to change the default name of your Playwright world from "RoleModellers", set `com.rolemodellers.rim.worldName=worldName`.
2. `com.rolemodellers.rim.bossEmail` and `com.rolemodellers.rim.bossPassword`, to change the default username and password for the Boss user from the [standard values](#).
3. `log4j.configuration`, if you wish to customise the logging carried out by RADRunner and the application server does not itself use log4j (see [Logging](#))

If you are using Jini, you *may need to set further properties for Java 2 Security*:

1. If Java 2 security is not enabled by default in the application server, set `java.security.manager` (no value required).
2. `java.security.policy=policyFileLocation` (a file path relative to the working directory of the application server). If the security policy file is placed under `\\myServer\rim` as suggested above, set this working directory to `\\myServer\rim`, and `policyFileLocation` to the name of the policy file created (e.g., `rm.policy`).
3. In certain application servers, other properties may also be required to enable Java 2 security - for instance, with WebSphere it is also necessary to specify `enableJava2Security=true` and `was.home=WebSphereInstallationRoot`.

With regard to these latter security properties, note that in certain application servers it is necessary to specify security settings in the command line that starts up the application server JVM, rather than in the EAR or via an administration console. With JBoss, for example, a line such as the following could be added to the standard bin directory file `run.bat`:

```
set          JAVA_OPTS=%JAVA_OPTS%          -Djava.security.manager
-Djava.security.policy=\\serverName\rim\rm.policy
-Dcom.rolemodellers.jini.codeBase=http://serverName:serverPort/jini
-Dcom.rolemodellers.jini.codeHost=serverName
-Dcom.rolemodellers.jini.worldHost=serverName
```

Such a line should be executed just before the application server is started - for instance, in the JBoss `run.bat` file it should be added just before the line:

```
%JAVA% %JAVA_OPTS% -classpath "%JBOSS_CLASSPATH%" org.jboss.Main
```

## Installation

%ARGS%

The *remaining system properties* found in *com/rolemodellers/openapi/RMRimProperties.properties* are optional, and control a variety of settings ranging from the transaction timeout to the number of items displayed at a time in different web pages:

```
# RADRunner JVM System Properties File

# The default configuration is a Windows installation with database persistence on JBoss
# You may need to over-ride, change, enable, or disable the following properties before

# File system locations for Role-Interaction Machine I/O
com.rolemodellers.rim.logPath=C:\\radrunner\\rim\\log\\
com.rolemodellers.rim.sessionPath=C:\\radrunner\\rim\\session\\
com.rolemodellers.rim.outPath=C:\\radrunner\\rim\\out\\
com.rolemodellers.rim.inPath=C:\\radrunner\\rim\\in\\

# The URL prefixes by which these locations can be accessed via HTTP -
# replace "localhost:8090/rim" with your host name, port and path
com.rolemodellers.rim.logUrlPrefix=http://localhost:8090/rim/log/
com.rolemodellers.rim.sessionUrlPrefix=http://localhost:8090/rim/session/
com.rolemodellers.rim.outUrlPrefix=http://localhost:8090/rim/out/
com.rolemodellers.rim.inUrlPrefix=http://localhost:8090/rim/in/

# Snapshot access protocol
com.rolemodellers.rim.snapshotProtocol=file

# Persistence - database
com.rolemodellers.rim.persistenceMechanism=Jdbc
com.rolemodellers.jdbc.useInternalConnectionPool=true
com.rolemodellers.jdbc.username=
com.rolemodellers.jdbc.password=
#MySQL
com.rolemodellers.jdbc.driver=com.mysql.jdbc.Driver
com.rolemodellers.jdbc.url=jdbc:mysql://localhost/radrunner
com.rolemodellers.jdbc.useClob=false
com.rolemodellers.jdbc.maxPlaywrightLength=0
com.rolemodellers.jdbc.useCreateTableAppendText=true
com.rolemodellers.jdbc.createTableAppendText=Type = InnoDB
#DB2
#com.rolemodellers.jdbc.driver=COM.ibm.db2.jdbc.net.DB2Driver
#com.rolemodellers.jdbc.url=jdbc:db2://SERVER_NAME/radrunner
#com.rolemodellers.jdbc.useClob=true
#com.rolemodellers.jdbc.maxPlaywrightLength=1000000
#com.rolemodellers.jdbc.useCreateTableAppendText=false
#com.rolemodellers.jdbc.createTableAppendText=

# Persistence - Jini
#com.rolemodellers.rim.persistenceMechanism=Javaspaces
#com.rolemodellers.jini.codeBase=http://localhost/jini
#com.rolemodellers.jini.codeHost=localhost
```

```

#com.rolemodellers.jini.worldHost=localhost

# Java 2 Security - used with Jini
java.security.manager=
java.security.policy=C:\\JBoss\\server\\default\\conf\\server.policy
# Java 2 Security - WebSphere only
enableJava2Security=true
was.home=C:/WebSphere/AppServer

# JNDI - JBoss 3.0.0
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.provider.url=jnp://localhost:1099
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
# JNDI - WebSphere 4.0.3 and 4.0.4
#javax.naming.Context.INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory

# HTTP Proxy
proxySet=false
#proxyHost=192.168.0.1
#proxyPort=6588

# Email - set these properties to enable sending email from a Role
mail.host=validSmtphost
user.name=validSmtphostUser

# Playwright World
com.rolemodellers.rim.worldName=RoleModellers
com.rolemodellers.rim.bossEmail=boss@rolemodellers.com
com.rolemodellers.rim.bossPassword=byron
com.rolemodellers.rim.automationInterval=60000

# Commercial licensing - leave blank for evaluation
com.rolemodellers.rim.licensedActors=
com.rolemodellers.rim.licenseKey=

# Logging
log4j.configuration=log4j.xml
com.rolemodellers.rim.logSuffix=csv
com.rolemodellers.rim.logger=com.rolemodellers.message.RMLog4JLogger
# Comma-Separated Variable [default] - for import into spreadsheet or database
com.rolemodellers.rim.logPattern=%-6r,%d{ISO8601},%15.15t,%-5p,%40.40c{3},%m%n
com.rolemodellers.rim.logThreadPattern={0},{1},
# Human readable
#com.rolemodellers.rim.logPattern=%-6r %d{ISO8601} [%15.15t] %-5p %40.40c{3} - %m%n
#com.rolemodellers.rim.logThreadPattern='{'{0}':{1}'}'

# You may need to change the port number to that used by your application server
com.rolemodellers.rim.mappingModel=http://localhost:8080/radrunner/castor/mappingModel.

# Locale
com.rolemodellers.rim.language=en
com.rolemodellers.rim.country=GB

# Do you want to see all Action steps on screens, including the automated ones?

```

## Installation

```
com.rolemodellers.rim.showAutomatedSteps=false

# The maximum size in bytes of files uploaded to the server
com.rolemodellers.rim.maximumUploadSize=1000000

# The properties in the next set are usually left alone

com.rolemodellers.rim.usePseudoSessions=false
com.rolemodellers.rim.sessionTimeout=1200000
com.rolemodellers.rim.nestedSeparator=.
com.rolemodellers.rim.persistenceTimeout=1000
com.rolemodellers.rim.bossName=RADRunner Boss Actor
com.rolemodellers.rim.fortyTwoObjectName=RADRunner 42 Role
com.rolemodellers.rim.fortyTwoClassPropertyName=42 Type
com.rolemodellers.rim.fortyTwoBossPropertyName=Boss
com.rolemodellers.rim.fortyTwoInstancePropertyName=42
com.rolemodellers.rim.roleBatchSize=8
com.rolemodellers.rim.actionBatchSize=8
com.rolemodellers.rim.actionStepBatchSize=8
com.rolemodellers.rim.playwrightSuffix=xml
com.rolemodellers.rim.dateInputFormat=dd.MM.yyyy HH:mm:ss
com.rolemodellers.rim.countSymbol=#
com.rolemodellers.rim.sumSymbol=+
com.rolemodellers.rim.cacheStylesheets=true
com.rolemodellers.jdbc.connectionTimeout=60000
com.rolemodellers.jdbc.connectionPoolInitialSize=10
com.rolemodellers.jdbc.connectionPoolReapInterval=300000
com.rolemodellers.jdbc.transactionManagerReapInterval=300000
com.rolemodellers.jdbc.table=RM_PLAYWRIGHT_OBJECT
com.rolemodellers.jdbc.columnPrefix=RMPO_
com.rolemodellers.jdbc.worldColumn=WORLD
com.rolemodellers.jdbc.idColumn=ID
com.rolemodellers.jdbc.nameColumn=NAME
com.rolemodellers.jdbc.classIdColumn=CLASS_ID
com.rolemodellers.jdbc.classNameColumn=CLASS_NAME
com.rolemodellers.jdbc.superclassIdColumn=SUPERCLASS_ID
com.rolemodellers.jdbc.superclassNameColumn=SUPERCLASS_NAME
com.rolemodellers.jdbc.emailColumn=EMAIL
com.rolemodellers.jdbc.passwordColumn=PASSWORD
com.rolemodellers.jdbc.typeColumn=TYPE
com.rolemodellers.jdbc.playwrightColumn=PLAYWRIGHT
com.rolemodellers.jdbc.playwrightColumnType=LONGTEXT
com.rolemodellers.jdbc.createdColumn=CREATED
com.rolemodellers.jdbc.lastUpdatedColumn=LAST_UPDATED
com.rolemodellers.jdbc.maxWorldLength=100
com.rolemodellers.jdbc.maxNameLength=100
com.rolemodellers.jdbc.maxEmailLength=100
com.rolemodellers.jdbc.maxPasswordLength=100
com.rolemodellers.jdbc.escapeCharacter=\
com.rolemodellers.jdbc.useEscapeProcessing=false
com.rolemodellers.jdbc.defaultTableXmlElementName=rows
com.rolemodellers.jdbc.rowXmlElementNameFormat=row{0,number,#}

# You should not need to change the following properties
```

```

com.rolemodellers.rim.loopLimit=1000000
com.rolemodellers.rim.generatedNameFormat={0} {1,number,#}
com.rolemodellers.rim.descriptionFormat={0} in {1}
com.rolemodellers.rim.useActionSteps=true
com.rolemodellers.rim.maintainRoleBoundary=false
com.rolemodellers.rim.idSeparator=:
com.rolemodellers.ejb.localContext=comp/env/ejb
com.rolemodellers.ejb.globalContext=comp/env/com/rolemodellers/ejb
com.rolemodellers.jini.transactionTimeout=1200000
com.rolemodellers.jini.stressCheckInterval=0
com.rolemodellers.jini.stressLimit=100
com.rolemodellers.jini.stressDelay=500
com.rolemodellers.jini.javaspaceFetchSize=1000
com.rolemodellers.jdbc.statementTimeout=10
com.rolemodellers.jdbc.dataSourceReference=jdbc/RMDB
com.rolemodellers.jdbc.maxTypeLength=100
com.rolemodellers.jdbc.dataSource=com/rolemodellers/jdbc/RMDB
com.rolemodellers.xml.exportFilenameFormat={0}_{1}_{2}.xml
com.rolemodellers.xml.schemaPrefix=xsd
com.rolemodellers.xml.schemaUri=http://www.w3.org/2001/XMLSchema
com.rolemodellers.xml.schemaInstancePrefix=xsi
com.rolemodellers.xml.schemaInstanceUri=http://www.w3.org/2001/XMLSchema-instance
com.rolemodellers.xml.schemaInstanceType=type
com.rolemodellers.xml.attribute=@
com.rolemodellers.xml.replaceWhitespaces=_

# Database drivers

com.rolemodellers.jdbc.mysql=com.mysql.jdbc.Driver

```

## 7. Deployment

Once Java 2 Security has been enabled (only necessary if using Jini), and the necessary properties have been set, deploy *radrunner.ear* to the application server.

Under JBoss, all that is necessary is to place the EAR in the *deploy* directory.

Under WebSphere:

1. Ensure that module visibility is set to "Application", so that the web and EJB modules in the application can see each other's classes, and also that these classes take precedence over any similarly named classes on the application server's classpath.
2. You can leave all EJB resource references with their default mappings (e.g., *SampleDataSource*).
3. You will be prompted "Application code necessary for installation has already been generated. Regenerate code now?" - you may reply "No", although no harm will be done if you reply "Yes".

Then [test that RADRunner works](#).

## **8. Getting Started**

You can access the RADRunner login page by entering into a browser the following URL:

*http://<application server host>:<application server port>/radrunner/RMLogin.jsp*

or the admin console via:

*http://<application server host>:<application server port>/radrunner*

The admin console is of limited functionality since most RADRunner maintenance is done under custom process control, from within the application itself.

Once RADRunner is up and running, for security reasons the first thing that should be done is to:

1. Login to the Boss user and 42 Role instance via the default email (*boss@rolemodellers.com*) and password (*byron*)
2. Change this email and password via *Maintain Users/Edit User*.

Then you can [start building processes](#).