JasperReports Server Community Project Source Build Guide

Release 5.0

http://community.jaspersoft.com

This is version 1212-JSP50-23 of the *JasperReports Server Source Build Guide*.

Table of Contents

CHAPTER 1 INTRODUCTION

JasperReports Server builds on JasperReports as a comprehensive family of Business Intelligence (BI) products, providing robust static and interactive reporting, report server, and data analysis capabilities. These capabilities are available as either stand-alone products, or as part of an integrated end-to-end BI suite utilizing common metadata and providing shared services, such as security, a repository, and scheduling. The server exposes comprehensive public integration interfaces enabling seamless integration with other applications and the capability to easily add custom functionality.

This guide assists such developers in obtaining, setting up, building, and running JasperReports Server from its source files.

Jaspersoft provides several other source of information to help extend your knowledge of JasperReports Server:

- Our Ultimate Guides document advanced features and configuration. They also include best practice recommendations and numerous examples. The guides are available as downloadable PDFs. Community project users can purchase individual guides or bundled documentation packs from the Jaspersoft online store. Commercial customers can download them freely from the support portal.
- Our free Business Intelligence Tutorials let you learn at your own pace, and cover topics for developers, system administrators, business users, and data integration users. The tutorials are available online from Professional Services section of our website.
- Our free samples, which are installed with JasperReports, iReport, and JasperReports Server, are documented online. The samples documentation can be found on our community website.

This document describes how to build from a command line shell under Linux or Windows. It does not address the process of building within an IDE (Integrated Development Environment) such as Eclipse.

1.1 Supported Build Configurations

The following table lists the target configurations that can be built from the source:

Application Server	Database
Tomcat, GlassFish, or JBoss	PostgreSQL
	MySQL

1.2 JasperReports Server Source Code Archives

The following table lists the source code archive files for JasperReports Server:

File	Description	Documented In
jasperreports-server-cp-5.0.0-src.zip	JasperReports Server source code	**Chapter 2** **Chapter 3**
jasperjpivot-5.0.0-src.zip	JasperJPivot source code	Appendix **A.1**
jasperserver-portlet-<version>-src.zip	JasperReports Server Portlet source code	Appendix **A.2**

This document refers to the location where you unpack these archive files as <js-src>.

CHAPTER 2 COMPONENTS REQUIRED FOR SOURCE BUILD

The components and versions listed in this section are required in order to build and run JasperReports Server:

- **Checking Your Java JDK**
- **Installing Maven**
- **Checking Your Application Server**
- **Checking Your Database Instance**

2.1 Checking Your Java JDK

The JasperReports Server source code can be compiled under Java 1.6 or 1.7. JasperReports Server does not run with versions of Java earlier than 1.6.

Note: As of release 4.5.0 Java 1.5 is no longer supported.

To check the version of your JDK (Java Development Kit), run the following command:

```
javac -version
```

To download the Java JDK, follow the instructions found at the Java web site: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

The Oracle/Sun JDK is the certified Java platform for JasperReports Server. This source build procedure has been specifically tested with the Oracle/Sun JDK. Additionally, JasperReports Server has been tested with OpenJDK 1.6.

2.2 Installing Maven

Apache Maven is used to compile, build, and package the JasperReports Server source code. The JasperReports Server development team uses Maven because of its capability to manage third party tool dependencies via remote, online repositories. Third party tools are typically packaged as Java archive files (JARs). For more information about Maven, see: http://maven.apache.org

You can download and install Maven from the Maven website: http://maven.apache.org/download.html#installation

Put the maven binary (mvn or mvn.exe) in your environment PATH so that you can execute mvn from the command line. To check your Maven version, run the following command:

```
mvn -version
```

If the maven executable is not in maven_home, add maven.home as a property in default_master.properties. maven.home = /usr/share/maven. This might be necessary when maven is installed via a package manager.

The JasperReports Server source code has been test-built with Maven version 3.0.4. For more information about supported Maven versions, see section **C.3, "Maven Troubleshooting," on page 30**.

2.3 Checking Your Application Server

To run JasperReports Server, you need an application server on the same computer as JasperReports Server. Stop the application server during the build and installation procedures.

The application servers supported by this build procedure are listed in section **1.1, "Supported Build Configurations," on page 2**. The Professional version of JasperReports Server runs under additional application servers. For information about all supported application servers, see the *JasperReports Server Installation Guide*.

2.4 Checking Your Database Instance

To run JasperReports Server, you need a database instance. Run the database server during the installation and build procedures.

The databases supported by this build procedure are listed in section **1.1, "Supported Build Configurations," on page 2**.

Chapter 3 Building JasperReports Server Source Code

The following sections include complete instructions for building the JasperReports Server source code.

This document describes how to build from a command line shell under Linux or Windows. It does not address the process of building within an IDE (Integrated Development Environment) such as Eclipse.

3.1 Introduction to Buildomatic Source Build Scripts

The JasperReports Server source code comes with a set of configuration and build scripts based on Apache Ant and known as the buildomatic scripts. These scripts are found in the following directory:

<js-src>/jasperserver/buildomatic

The buildomatic scripts automate most aspects of the configuring, building, and deploying the source code. Apache Ant is bundled into the source code distribution to simplify the setup.

3.2 Downloading and Unpacking JasperReports Server Source Code

3.2.1 Downloading the Source Archive

Download source code for the commercial version of JasperReports Server from the Jaspersoft Community site: http://community.jaspersoft.com/download

File to download: jasperreports-server-cp-5.0.0-src.zip

3.2.2 Unpacking the Source Archive

Unpack the jasperreports-server-cp-5.0.0-src.zip file to a directory location, such as C:\ or /home/<user>. The resulting location is referred to as <js-src> in this document.

Windows:	<js-src> example is C:\jasperreports-server-cp-5.0.0-src
Linux:	<js-src> example is /home/<user>/jasperreports-server-5.0.0-src

3.2.3 Source Code Package Structure

After unpacking, the source directory has the following structure:

Directory	Description
<js-src>/apache-ant	Bundled version of Apache Ant
<js-src>/jasperserver	JasperReports Server open source code for general features
<js-src>/jasperserver-repo	Dependent jar files (not readily available publicly)

3.3 Checking Apache Ant

The Apache Ant tool is bundled (pre-integrated) into the source code distribution package so you do not need to download or install Ant in order to run the buildomatic scripts. Included shell scripts for Windows and Linux are pre-configured to use the bundled version of Apache Ant. Call these scripts from the command line in this manner:

```
js-ant <target-name>   or
./js-ant <target-name>
```

3.3.1 Using Your Own Apache Ant: Get ant-contrib.jar

Alternatively, if you prefer to use your own version of Apache Ant, get the file ant-contrib-<ver>.jar. This JAR enables conditional logic in Ant scripts.

1. Make sure you are using Apache Ant 1.8.1 or higher.
2. Copy the file ant-contrib-1.0b3.jar from the <js-src>/apache-ant/lib directory to your <ant-home>/lib directory:

   From:

   <js-src>/apache-ant/lib/ant-contrib.jar   or

   <js-src>/jasperserver/buildomatic/extra-jars/ant-contrib.jar

   To:

<ant-home>/lib	(General example)
C:\apache-ant-1.8.1\lib	(Windows example)
/usr/share/java/apache-ant/lib	(Linux example)
/usr/share/ant/lib	(Mac example)

3.4 Configuring the Buildomatic Properties

The buildomatic scripts are found at the following location:

```
<js-src>/jasperserver/buildomatic
```

The buildomatic scripts are used to build JasperReports Server source code and to configure proper settings for supported application servers and databases. The main file for configuring these settings is default_master.properties, which is a Java properties file.

In Java properties files, backslashes in Windows paths can be escaped with a second backslash (\\).

When using Apache Ant, the single forward slash (/) also works on most Windows systems.

1. Configure the default_master.properties file.

   The default_master.properties file specifies the type of database you are using, the connect information for the database, and the location of your application server. The source distribution includes a properties file that is specific to each type of database. You add your specific settings to this file and save it as your default_master.properties file. Use a procedure in one of the following sections, depending on your database type.
2. Refresh your buildomatic settings.

   To clean and regenerate buildomatic script settings after changing your default_master.properties file, run Ant with these targets:

Commands	Description

`js-ant clean-config`
`js-ant gen-config`
	Clears the buildomatic/build_conf/default directory.
	Rebuilds the configuration settings

In addition, any time you modify the default_master.properties, configuration settings get automatically re-generated in the buildomatic/build_conf directory.

3.4.1 PostgreSQL

1. Go to the buildomatic directory in the source distribution:

   ```
   cd <js-src>/jasperserver/buildomatic
   ```
2. Copy the appropriate file to the current directory and change its name at the same time:

Windows:	`copy sample_conf\source\postgresql_master.properties default_master.properties`
Linux:	`cp sample_conf/source/postgresql_master.properties default_master.properties`

3. Edit the new default_master.properties file and set the following properties to your local settings:

Property	Examples
`appServerType`	`appServerType=tomcat7 (tomcat5/6, jboss, glassfish2/3, skipAppServerCheck)`

`appServerDir`	`appServerDir = C:\\Program Files\\Apache Software Foundation\\Tomcat 7.0`
	`appServerDir = /home/user/apache-tomcat-7.0.26`

`dbHost`	`dbHost=localhost`
`dbUsername`	`dbUsername=postgres`
`dbPassword`	`dbPassword=postgres`

`maven`	`maven = C:\\apache-maven-3.0.4\\bin\\mvn`
	`maven = /usr/bin/mvn`

`js-path`	`js-path = C:\\jasperreports-server-cp-5.0.0-src\\jasperserver`
	`js-path = /home/<user>/jasperreports-server-cp-5.0.0-src/jasperserver`

`js-pro-path`	`Set to an existing folder, which is not used, but must be set`

`repo-path`	`repo-path = C:\\jasperreports-server-cp-5.0.0-src\\jasperserver-repo`
	`repo-path = /home/<user>/jasperreports-server-cp-5.0.0-src/ jasperserver-repo`

4. Refresh your buildomatic settings.

3.4.2 MySQL

As of release 4.2.0, the source packaging no longer contains a MySQL JDBC driver in the buildomatic folder tree. Therefore, in order to complete the source build steps (which includes creating the jasperserver database), download a MySQL JDBC driver.

1. Download the JDBC driver, mysql-connector-java-5.1.17-bin.jar or later from this web site:

   http://dev.mysql.com/downloads/connector/j/

   Place the MySQL driver in <js-src>/jasperserver/buildomatic/conf_source/db/mysql/jdbc

2. Go to the buildomatic directory in the source distribution:

```
cd <js-src>/jasperserver/buildomatic
```

3. Copy the appropriate file to the current directory and change its name at the same time:

Windows: `copy sample_conf\source\mysql_master.properties default_master.properties`

Linux: `cp sample_conf/source/mysql_master.properties default_master.properties`

4. Edit the new default_master.properties file and set the following properties to your local settings:

Property	Examples
`appServerType`	`appServerType = tomcat7 (or tomcat5/6, jboss, or glassfish2/3)`

`appServerDir`	`appServerDir = C:\\Program Files\\Apache Software Foundation\\Tomcat 7.0`
	`appServerDir = /home/user/apache-tomcat-7.0.26`

`dbHost`	`dbHost = localhost`
`dbUsername`	`dbUsername = root`
`dbPassword`	`dbPassword = password`
`maven.jdbc.groupId`	`maven.jdbc.groupId = mysql`
`maven.jdbc.artifactId`	`maven.jdbc.artifactId = mysql-connector-java`
`maven.jdbc.version`	`maven.jdbc.version = 5.1.17-bin`

`maven`	`maven = C:\\apache-maven-3.0.4\\bin\\mvn`
	`maven = /usr/bin/mvn`

`js-path`	`js-path = C:\\jasperreports-server-cp-5.0.0-src\\jasperserver`
	`js-path = /home/<user>/jasperreports-server-cp-5.0.0-src/jasperserver`

`js-pro-path`	`Set to an existing folder, which is not used, but must be set`

`repo-path`	`repo-path = C:\\jasperreports-server-cp-5.0.0-src\\jasperserver-repo`
	`repo-path = /home/<user>/jasperreports-server-cp-5.0.0-src/jasperserver-repo`

5. Refresh your buildomatic settings.

3.5 Building JasperReports Server

Now that the default_master.properties file has been edited, you can start building the JasperReports Server source code. Make sure that your database server is running and that your application server is stopped (unless it's GlassFish, which should be running). After you execute the first build target, the buildomatic scripts automatically configure the necessary properties and store these settings in the following directory:

<js-src>/jasperserver/buildomatic/build_conf/default

After executing each Ant target below, look for the message BUILD SUCCESSFUL.

To build JasperReports Server:

1. Modify default_master.properties to match your environment. For more information, see **3.4, "Configuring the Buildomatic Properties," on page 7**.
2. Start the database server.
3. Stop the application server unless it's GlassFish, which should be running.
4. Run the commands listed in **Table 3-1**.

   After executing each Ant target in **Table 3-1**, look for the message BUILD SUCCESSFUL.

Table 3-1 Commands for Building JasperReports Server

Commands	Description
`cd <js-src>/jasperserver/buildomatic`	
`js-ant build-ce`	Builds the Community Project source code.
`js-ant create-load-all-dbs-ce`	◆ Creates and loads the jasperserver database. ◆ Imports core bootstrap resources into the jasperserver repository. ◆ Creates and loads sample databases. ◆ Imports sample resources into the jasperserver repository.
or	
`js-ant create-load-js-db-ce`	Do not run this command if you ran `create-load-all-dbs-pro`. ◆ Creates and loads the jasperserver database. ◆ Imports core bootstrap resources into the jasperserver repository. ◆ **Does not load** sample data or sample databases.
`js-ant deploy-webapp-ce`	Deploys JasperReports Server to the application server.

3.5.1 Detailed Description of the deploy-webapp-ce Target

The deploy-webapp-ce target carries out the following actions in your application server environment:

- Deletes any existing jasperserver WAR file.
- Copies the JDBC driver to the appropriate application server directory.
- Adds a data source definition to the appropriate application server directory.
- Deploys the newly built jasperserver WAR file.
- Deletes files within the application server work directory (to clear out compiled JSP files and other cached files).
- Under Tomcat, delete the old version of <tomcat>/conf/Catalina/Localhost/jasperserver.xml if present.

3.5.2 Running Ant in Debug Mode

Ant can be run with a -v (verbose) or a -d (debug) option to help with troubleshooting, for example:

```
js-ant -v build-ce
```

3.6 Running Integration-Tests (Optional)

After you successfully build the source code, you can choose to run the integration-tests. Currently, running the integration-tests requires that you drop and recreate the `jasperserver` database before executing the tests.

Note: You could get an error with the integation-tests if you do not have a jasperserver.license in your <home> folder.

To run integration-tests:

1. Make the buildomatic directory your current directory:
   ```
   cd <js-svc>/jasperserver/buildomatic
   ```
2. Enter these commands:
   ```
   js-ant add-jdbc-driver
   js-ant drop-js-db
   js-ant create-js-db
   js-ant init-js-db-ce
   js-ant run-integration-tests-ce
   ```

3.6.1 Setting Java JVM Options

This step is required in order to run JasperReport Server. Otherwise, you could get an an error regarding Permgen memory.

For JasperReports Server to run effectively, you must increase the Java JVM runtime memory options. For more information, including the JAVA_OPTS settings, refer to **Appendix B, "Java JVM Settings," on page 27**.

3.7 Starting Your Application Server

You can now start your application server, or restart GlassFish. Your database should already be running.

3.8 Logging into JasperReports Server

You can now login to JasperReports Server through a web browser. If you specified all the default values when setting up JasperReports Server, log in as follows:

Enter the login URL with the default port number:

http://localhost:8080/jasperserver

Log into JasperReports Server as superuser or jasperadmin:

User ID: `jasperadmin` Password: `jasperadmin`

After logging into JasperReports Server, you can create reports, run reports, and create dashboards. Refer to the *JasperReports Server User Guide* for more information about the application.

If you are unable to login or have other problems, refer to **Appendix C, "Troubleshooting," on page 29**, or refer to the *JasperReports Server Installation Guide*, which provides additional troubleshooting information.

3.9 JasperReports Server Log Files

The JasperReports Server runtime log is written to the following Tomcat location (for example):

<tomcat>/webapps/jasperserver/WEB-INF/logs/jasperserver.log

The log4j logging level can be controlled by configuring the log4j.properties file in the following location:

<tomcat>/webapps/jasperserver/WEB-INF/log4j.properties

CHAPTER 4 ADDITIONAL BUILDOMATIC INFORMATION

The Ant-based buildomatic scripts contain support files for the setup and configuration of a number of databases and application servers. This chapter gives the locations of many of these files.

This chapter also contains details about the `create-load-all-dbs-ce` and `create-load-js-db-ce` targets used in building the JasperReports Server source code using the simplified targets added in Release 4.5.

4.1 Generated Property Files

After you set your database and application server property values, you run buildomatic scripts to generate the database and application server configuration files to run JasperReports Server. Generated property files are in the following directory:

<js-src>/jasperserver/buildomatic/build_conf/default

Some of the key configuration files are:

js.jdbc.properties

js.quartz.properties

js-jboss-ds.xml

maven_settings.xml - (This is the maven settings file used by the source build)

More generated property files are in the following directory:

<js-src>/jasperserver/buildomatic/build_conf/default/webapp

Some of the configuration files in this directory are:

META-INF/context.xml

WEB-INF/hibernate.properties

WEB-INF/js.quartz.properties

Running `clean-config` removes these generated files. Running `gen-config` or any other target, regenerates these files.

4.2 Existing and Generated Database SQL Files

Buildomatic files that support various databases are located in:

<js-src>/jasperserver/buildomatic/install_resources/sql/<db-type>/

The source code build creates the JasperReports Server repository database schema using these files from the database directory:

js-ce-create.ddl

js-ce-drop.ddl

When the buildomatic target `create-js-ddl-pro` is run, these database files are freshly generated for your specified database platform. The files are generated to the following location:

<js-src>/jasperserver/repository-hibernate/build-db/target/sql/

Then, the files are automatically copied into their buildomatic directory location:

<js-src>/jasperserver/buildomatic/install_resources/sql/<db-type>/

These generated files overwrite the ones already existing in the buildomatic directory location.

4.3 Generated WAR File Location and deploy-webapp-ce Target

The JasperReports Server source code build creates a jasperserver WAR file. The build assembles the WAR file into the following location:

<js-src>/jasperserver/jasperserver-war/target

When the `build-ce` target is run, buildomatic finishes creating the jasperserver WAR file, and copies the file to this location for use by subsequent buildomatic targets:

<js-src>/jasperserver/buildomatic/install_resources/war/jasperserver

Later, when you run the buildomatic target `deploy-webapp-ce`, the following actions take place under Tomcat, for example:

Files:	<js-src>/jasperserver/buildomatic/install_resources/war/jasperserver/*
Copied to:	<tomcat>/webapps
File:	<js-src>/jasperserver/buildomatic/build_conf/default/webapp/META-INF/context.xml
Copied to:	<tomcat>/webapps/jasperserver/jasperserver/META-INF
Files:	<js-src>/jasperserver/buildomatic/build_conf/default/webapp/WEB-INF/hibernate.properties <js-src>/jasperserver/buildomatic/build_conf/default/webapp/WEB-INF/js.quartz.properties
Copied to:	<tomcat>/webapps/jasperserver/WEB-INF
File:	<js-src>/jasperserver/buildomatic/build_conf/db/postgresql/jdbc/postgresql-9.0-801.jdbc3.jar
Copied to:	<tomcat>/lib

4.4 Details on New Build Targets

As of Release 4.5, new targets consolidate and simplify the handling of the `jasperserver` and sample databases. These targets, specified in **Chapter 3, "Building JasperReports Server Source Code," on page 5** are:

- `create-load-js-db-ce`
- `create-load-all-dbs-ce`

4.4.1 create-load-js-db-ce

This buildomatic target is a consolidation of the following original targets:

- (`drop-js-db`, if necessary)
- `create-js-db`
- `init-js-db-ce`
- `import-minimal-ce`

Additionally, functionality has been added to check whether or not the `jasperserver` database already exists. If the database already exists, then a command line prompt asks the user whether or not to delete and re-create the database.

4.4.2 create-load-all-dbs-ce

This buildomatic target is a consolidation of the following original targets:

- (`drop-js-db`, if necessary)
- `create-js-db`
- `init-js-db-ce`
- `import-minimal-ce`
- `import-sample-data-ce`
- (`drop-foodmart-db`, if necessary)
- `create-foodmart-db`
- `load-foodmart-db`
- (`drop-sugarcrm-db`, if necessary)
- `create-sugarcrm-db`
- `load-sugarcrm-db`

Additionally, functionality has been added to check whether or not the `jasperserver` database already exists. If the database already exists, then a command line prompt asks the user whether or not to delete and re-create the database. The same logic is used for the sample databases: foodmart and sugarcrm.

4.4.3 Create Database Schema

The new consolidated database scripts do not regenerate the database schema. Instead, the existing, default database schema files are used. To regenerate the database schema files, run the following target:

```
js-ant build-js-ddl-ce
```

The files are generated to the following location:

<js-src>/jasperserver/repository-hibernate/build-db/target/sql/

Then, the files are automatically copied into their buildomatic directory location:

<js-src>/jasperserver/buildomatic/install_resources/sql/<db-type>/

4.5 Older Buildomatic Commands

This section describes the method for building JasperReports Server that was in place for release 4.2 and earlier.

The recommended way to build the JasperReports Server source code is to use the buildomatic scripts as described in **Chapter 3, "Building JasperReports Server Source Code," on page 5**. You don't have to type as many commands.

To build JasperReports Server using older Buildomatic commands:

1. Edit the `default_master.properties` file for your particular environment.
2. Start the database server.
3. Stop the application server (unless it's GlassFish, which should be running).

After you execute the first build target, the buildomatic scripts automatically configure the necessary properties and store these settings in the following directory:

```
<js-src>/jasperserver/buildomatic/build_conf/default
```

After executing each Ant target below, look for the message `BUILD SUCCESSFUL.`

4. Execute the following steps at the command line:

Commands	**Description**
`cd <js-src>/jasperserver/buildomatic`	
`js-ant add-jdbc-driver`	Installs the JDBC driver to mvn local repository
`js-ant build-ce`	Builds the Community Project source code
`js-ant create-js-db`	If the jasperserver database already exists, first run `js-ant drop-js-db`
`js-ant create-sugarcrm-db`	Creates sample data for integration-tests
`js-ant load-sugarcrm-db`	
`js-ant create-foodmart-db`	Creates sample data for integration-tests
`js-ant load-foodmart-db`	Can run for 10 minutes or more
`js-ant build-js-ddl-ce`	Creates the database schema files for your database type
`js-ant init-js-db-ce`	Loads the schema into database
`js-ant run-integration-tests-ce`	Runs integration-tests (Optional)
`js-ant run-production-data-ce`	Put core bootstrap and Sample data into jasperserver db
`js-ant deploy-webapp-ce`	Deploys JasperReports Server to the application server

Chapter 5 Configuring the Build Environment Manually

The steps to configure the build environment manually describe how to use a Tomcat application server and a PostgreSQL database. Configuring the build environment for other application servers and databases is similar.

Begin setting up your build environment by downloading and unpacking the JasperReports Server source code package as described in section **3.2, "Downloading and Unpacking JasperReports Server Source Code," on page 5**.

5.1 Setting Up the JasperReports Server Build Manually

The following files are necessary to configure the build:

- settings.xml - Maven settings
- hibernate.cfg.xml - Hibernate build-time settings
- js.jdbc.properties - Database settings
- js.quartz.properties - Email server, scheduling, and quartz utility settings

Samples of these files are included in the source code package in the following directory:

<js-src>/jasperserver/scripts/dev-setup

In order to configure Maven, you must create your own version of these files in a directory named `.m2` within your home directory. To create this directory, do the following:

Windows XP:	`cd "\Documents and Settings\<user>"`
	`mkdir .m2`

Windows 7:	`cd \Users\<user>`

Linux:	`cd $HOME`
	`mkdir .m2`

The period (`.`) in `.m2` that indicates a hidden directory.

5.2 Setting the Maven Java Memory Option

When you run the JasperReports Server integration-tests or production-data creation, they can fail with an out of memory error.To set a larger JVM heap size, set the following environment variable in your shell environment:

Windows:	`set MAVEN_OPTS=-Xmx256m`
Linux:	`export MAVEN_OPTS=-Xmx256m`

5.3 Creating the settings.xml File

Settings.xml is the main configuration and properties setting file that is used by the Maven build tool.

The `settings.xml` file must reside directly within the `.m2` directory:

Windows XP:	`copy <js-src>\jasperserver\scripts\dev-setup\settings.xml "C:\Documents and Settings\<user>\.m2"`
Windows 7:	`copy <js-src>\jasperserver\scripts\dev-setup\settings.xml C:\Users\<user>\.m2`
Linux:	`cp <js-src>/jasperserver/scripts/dev-setup/settings.xml /home/<user>/.m2`

Maven uses the `settings.xml` file for all of the configuration options that affect the build.

If you use the buildomatic scripts to build JasperReports Server, all Maven settings (and other settings) are handled automatically. For more information, see **Chapter 3, "Building JasperReports Server Source Code," on page 5**.

Modify values in the `settings.xml` file to match your environment, for example on a Linux platform:

```
<test.hibernate.cfg>/home/<user>/.m2/hibernate.cfg.xml</test.hibernate.cfg>
<test.hibernate.jdbc.properties>/home/<user>/.m2/js.jdbc.properties
  </test.hibernate.jdbc.properties>

<js.quartz.properties>/home/<user>/.m2/js.quartz.properties</js.quartz.properties>
<js.quartz.script>/home/<user>/jasperreports-server-5.0.0-src/jasperserver/scripts/
quartz/tables_<database>.sql</js.quartz.script>

<repository>
  <id>jasperServer</id>
  <name>Base repository for Jasper Server</name>
  <url>file:///home/<user>/jasperreports-server-5.0.0-src/jasperserver-repo</url>
</repository>
```

Jaspersoft artifacts can be also retrieved from a central repository. For example, you can retrieve Jaspersoft artificats from the <js-path>/buildomatic/conf_source/templates/maven_settings_mirror.xml file.

5.4 Creating the hibernate.cfg.xml File

Copy the `hibernate.cfg.xml` file to your `.m2` directory:

Windows XP:	`copy <js-src>\jasperserver\scripts\dev-setup\hibernate.cfg.xml "C:\Documents and Settings\<user>\.m2"`
Windows 7:	`copy <js-src>\jasperserver\scripts\dev-setup\hibernate.cfg.xml C:\Users\<user>\.m2`
Linux:	`cp <js-src>/jasperserver/scripts/dev-setup/hibernate.cfg.xml /home/<user>/.m2`

Locate the following properties and modify the values in **bold** to match your own environment:

```
<property name="connection.driver_class">org.postgresql.Driver</property>
<property name="connection.url">jdbc:postgresql://localhost:5432/jasperserver</property>
<property name="connection.username">postgres</property>
<property name="connection.password">postgres</property>

<property name="dialect">org.hibernate.dialect.PostgresqlNoBlobDialect</property>
```

5.5 Creating the js.jdbc.properties File

Copy the `js.jdbc.properties` file to your `.m2` directory:

Windows XP:	`copy <js-src>\jasperserver\scripts\dev-setup\js.jdbc.properties "C:\Documents and Settings\<user>\.m2"`
Windows 7:	`copy <js-src>\jasperserver\scripts\dev-setup\js.jdbc.properties C:\Users\<user>\.m2`
Linux:	`cp <js-src>/jasperserver/scripts/dev-setup/js.jdbc.properties /home/<user>/.m2`

Locate the following properties and modify the values in **bold** to match your own environment:

```
metadata.hibernate.dialect=org.hibernate.dialect.PostgresqlNoBlobDialect

metadata.jdbc.driverClassName=org.postgresql.Driver
metadata.jdbc.url=jdbc:postgresql://localhost:5432/jasperserver
metadata.jdbc.username=postgres
metadata.jdbc.password=postgres
metadata.jdbc.database=jasperserver

metadata.jndi=jdbc/jasperserver

test.jdbc.driverClassName=org.postgresql.Driver
test.jdbc.url=jdbc:postgresql://localhost:5432/sugarcrm
test.jdbc.username=postgres
test.jdbc.password=postgres

test.jndi=jdbc/sugarcrm

foodmart.jdbc.driverClassName=org.postgresql.Driver
foodmart.jdbc.url=jdbc:postgresql://localhost:5432/foodmart
foodmart.jdbc.username=postgres
foodmart.jdbc.password=postgres

foodmart.jndi=jdbc/foodmart
```

5.5.1 Creating the js.quartz.properties File

Copy the `js.quartz.properties` file to your `.m2` directory:

Windows XP:	`copy <js-src>\jasperserver\scripts\dev-setup\js.quartz.properties "C:\Documents and Settings\<user>\.m2"`
Windows 7:	`copy <js-src>\jasperserver\scripts\dev-setup\js.quartz.properties C:\Users\<user>\.m2`
Linux:	`cp <js-src>/jasperserver/scripts/dev-setup/js.quartz.properties /home/<user>/.m2`

Locate the following properties and modify the values in **bold** to match your own environment.

```
report.scheduler.web.deployment.uri=http://localhost:8080/jasperserver
js.report.scheduler.mail.sender.host=mail.localhost
js.report.scheduler.mail.sender.port=25
js.report.scheduler.mail.sender.protocol=smtp
js.report.scheduler.mail.sender.username=admin
js.report.scheduler.mail.sender.password=password
js.report.scheduler.mail.sender.from=admin@localhost
```

5.6 Setting Up the JDBC Driver

The `<home>/.m2/settings.xml` file specifies a JDBC driver used to generate database specific schemas that are also needed to run integration-tests.

If you use PostgresSQL for your JasperReports Server repository database, then the `settings.xml` values are pre-set for the PostgresSQL JDBC driver. Maven looks for the PostgreSQL driver in the following location:

<home>/.m2/repository/postgresql/

where <home> is the Maven home directory.

5.7 Manual Creation of the JasperReports Server Databases

JasperReports Server runs with a repository database that is typically named `jasperserver`. The creation of the jasperserver database and, additionally, the sample databases is automatically handled by the automated buildomatic steps. If you would like to manually create your databases, here is an example with the PostgreSQL database.

5.7.1 Manually Creating Databases: PostgreSQL

The default database configuration used by the JasperReports Server source code uses these values:

Parameter	Default Value
Database Host Name	`localhost`
Database Port	`5432`
Database User Name	`postgres`
Database User Name (alternate: created by installer)	`jasperdb`
Database Password	`postgres`

1. To create the `jasperserver` database, log into PostgreSQL and create the databases:

```
cd <js-install>/buildomatic/install_resources/sql/postgresql

psql -U postgres -W
postgres=#create database jasperserver encoding='utf8';
postgres=#\c jasperserver;
postgres=#\i js-pro-create.ddl
postgres=#\i quartz.ddl
postgres=#\q
```

2. Run the following commands if you want to install sample databases:

```
cd <js-install>/buildomatic/install_resources/sql/postgresql

psql -U postgres -W
postgres=#create database sugarcrm encoding='utf8';
postgres=#create database foodmart encoding='utf8';
postgres=#\c sugarcrm;
postgres=#\i sugarcrm-postgresql.sql; (first make sure the file is unzipped)
postgres=#\c foodmart;
postgres=#\i foodmart-postqresql.sql; (first make sure the file is unzipped)
postgres=#\i supermart-update.sql;
postgres=#\q
```

5.7.2 Additional Databases

For information on manual setup of databases other than PostgreSQL, refer to the *JasperReports Server Installation Guide*.

5.8 Building the JasperReports Server Source Code Manually

To build the source code manually, Maven will download most of the 3rd party jar dependencies from external, remote repositories. To skip the download of these dependencies, you can use the contents of the jasperreports-server-cp-5.0.0-maven-repository.zip file.

```
cd <js-src>/jasperserver
mvn clean install
cd <js-src>/jasperserver/jasperserver-repository-hibernate/build-db
mvn clean install
cd <js-src>/jasperserver/production-tests                (Optional)
mvn clean install
cd <js-src>/jasperserver
mvn clean install
cd <js-src>/jasperserver/repository-hibernate/build-db
mvn clean install
cd <js-src>/jasperserver/production-tests             (Optional)
mvn clean install
```

The `mvn clean install` command that you executed in the jasperserver directory builds the jasperserver WAR file. If everything compiles cleanly, you can find this WAR file in the jasperserver/jasperserver-war/target directory.

If you encounter errors, refer to section **C.1, "Build Troubleshooting," on page 29** for help with debugging.

5.9 Copying the JDBC Driver JAR

Before running JasperReports Server, you must have the Apache Tomcat application server available and configured with the JDBC driver for JasperReports Server to use. Within Tomcat, JasperReports Server requires a JDBC driver to connect to its repository database.

Copy the file <js-src>/jasperserver/buildomatic/conf_source/db/postgresql/jdbc/postgresql-9.0-801.jdbc3.jar or postgresql-9.0-801.jdbc4.jar to the following location:

Tomcat 5.5:	<tomcat>/common/lib
Tomcat 6.0:	<tomcat>/lib
Tomcat 7.0	<tomcat>/lib

5.10 Validating Tomcat Related Configuration Files

5.10.1 Validating Context.xml

The JasperReports Server build process creates a `context.xml` file that Tomcat uses to connect to the database.

Verify that the following file was created with the database settings that you put into your `<home>/.m2/js.jdbc.properties` configuration file:

<js-src>/jasperserver/jasperserver-war/target/jasperserver/META-INF/context.xml

The PostgreSQL settings are similar to the following, but should have the correct information for your database setup; pay special attention to the user names, passwords, host names, and port numbers listed near the end of the file:

```
<Resource name="jdbc/jasperserver" auth="Container" type="javax.sql.DataSource"
  maxActive="100" maxIdle="30" maxWait="10000"
  username="postgres" password="postgres" driverClassName="org.Postgresql.Driver"
  url="jdbc:postgresql://localhost:5432/jasperserver"/>

<Resource name="jdbc/sugarcrm" auth="Container" type="javax.sql.DataSource"
  maxActive="100" maxIdle="30" maxWait="10000"
  username="postgres" password="postgres" driverClassName="org.Postgresql.Driver"
  url="jdbc:postgresql://localhost:5432/sugarcrm"/>

<Resource name="${foodmart.jndi}" auth="Container" type="javax.sql.DataSource"
  maxActive="100" maxIdle="30" maxWait="10000"
  username="postgres" password="postgres" driverClassName="org.Postgresql.Driver"
  url="jdbc:postgresql://localhost:5432/foodmart"/>
```

5.10.2 Validating Other Database Related Files

If you have errors, such as database failures, you can also check that you have the correct hibernate dialect setting:

<js-src>/jasperserver/jasperserver-war/target/jasperserver/WEB-INF/hibernate.properties

5.11 Copying the JasperReports Server WAR File to Tomcat

Now that the JasperReports Server source code has been built, you can manually deploy your jasperserver WAR file to your application server. Be sure to copy the entire directory:

Copy:	<js-src>/jasperserver/jasperserver-war/target/jasperserver/*
To:	<tomcat>/webapps

5.12 Starting JasperReports Server and Logging In

First, make sure that your database is running. You can now start your application server, which in turn starts JasperReports Server.

Enter the login URL with the default port number:

http://localhost:8080/jasperserver

Log in with credentials for a JasperReports Server system administrators:

- UserID: `jasperadmin`
- Password: `jasperadmin`.

APPENDIX A BUILDING OTHER JASPERREPORTS SERVER COMPONENTS

This appendix describes how to build other JasperReports Server components:

- **Building JasperJPivot Source Code**
- **Building and Running the JasperReports Server Portlet**

A.1 Building JasperJPivot Source Code

JasperJPivot is adapted from the JPivot open source project. It provides the web interface for Jaspersoft OLAP. In addition, JasperJPivot includes usability enhancements in the areas of navigation, configuration, and scalability.

The JasperJPivot build requires a local jasperserver-repo file. You can get the source code package from the Jaspersoft community website.

Download the source code package:

On the the Jaspersoft technical support website (login required).

Look for a file with the following name:

jasperjpivot-5.0.0-src.zip

Build the source code package:

Unpack the downloaded source code package zip file.

Next, follow the instructions found in the <unpacked-src>/Building-JasperJPivot-Source.pdf.

The process of building the JasperJPivot requires Apache Maven. For more information, see section **2.2, "Installing Maven," on page 3**.

A.2 Building and Running the JasperReports Server Portlet

The JasperReports Server portlet can be deployed to the Liferay Portal or to the JBoss Portal so that reports in the JasperReports Server repository can be displayed in your Portal environment.

Jaspersoft provides the source code for the JasperReports Server portlet so that developers can customize and extend the application for their specific needs. You can get the source code package from the Jaspersoft community website.

The process of building the JasperReports Server Portlet WAR file requires Apache Maven. For more information, see section **2.2, "Installing Maven," on page 3**.

Download the source code package:

On the the Jaspersoft technical support website (login required).

Look for a file with the following name:

JasperReportsServer-portlet-<ver>-src.zip

Build the source code package:

First, unpack the downloaded source code package zip file.

Next, follow the instructions in the Build Readme.txt file (found in the root unpacked folder).

Also, look for additional Readme.txt information in the <unpacked-folder>/docs directory.

In Jaspersoft 5.0, the Portlet build supports a new parameter, jasper.repo.url that points to a repository server. Rever to *JasperReportsServer Portlet Source Build README.txt* for details.

For instructions on deploying and running the JasperReports Server Portlet, refer to the *JasperReports Server Administrator Guide* and the readme files at the root of the unpacked zip file.

Appendix B Java JVM Settings

For additional information on Java settings, refer to the *JasperReports Server Installation Guide*.

B.1 Java 1.6 and 1.7

JasperReports Server is supported on Java 1.6 and 1.7. Java Virtual Machine (JVM) runtime parameters need to be correctly set to avoid conflicts with JasperReports Server's AXIS-based web service classes. These conflicts could cause web services and the resources that rely on them, such as XML/A connections, to fail.

The options you need and how you set them depends on your version of Java, your application server, and how it is deployed. In addition, there's a setting to support localization when running with an Oracle database.

The following tables give the recommended settings for Java 1.6 and 1.7. You can also copy these settings from the files in the <js-src>/jasperserver/scripts/java-settings directory.

The settings in this section apply specifically to the Oracle/Sun JVM. Other JVMs may or may not have equivalent settings.

B.1.1 Tomcat and JBoss JVM Options

The following tables present some typical settings of JVM options that affect JasperReports Server. For information about changing a JVM option setting for your particular environment, see your application server documentation.

JVM Options on Windows	
Options for Java 1.6 and 1.7	set JAVA_OPTS=%JAVA_OPTS% -Xms1024m -Xmx2048m -XX:PermSize=32m -XX:MaxPermSize=512m -Xss2m -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled
For Oracle	set JAVA_OPTS=%JAVA_OPTS% -Doracle.jdbc.defaultNChar=true
Additional options for Java 1.6-1.7 and JBoss	set JAVA_OPTS=%JAVA_OPTS% -Djavax.xml.soap.MessageFactory=org.apache.axis.soap.MessageFactoryImpl -Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis.soap.SOAPConnectionFactoryImpl -Djavax.xml.soap.SOAPFactory=org.apache.axis.soap.SOAPFactoryImpl -Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactoryImpl

JasperReports Server doesn’t provide a virtual X frame buffer on Linux. If your Linux applications are graphical, set the -Djava.awt.headless=true to prevent Java from trying to connect to an X-Server for image processing.

JVM Options on Linux and Mac OSX	
Options for Java 1.6 and 1.7	export JAVA_OPTS="$JAVA_OPTS -Xms1024m -Xmx2048m -XX:PermSize=32m -XX:MaxPermSize=512m -Xss2m -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled"
For Oracle	export JAVA_OPTS="$JAVA_OPTS -Doracle.jdbc.defaultNChar=true"
Additional options for Java 1.6-1.7 and JBoss 4.5	export JAVA_OPTS="$JAVA_OPTS -Djavax.xml.soap.MessageFactory=org.apache.axis.soap.MessageFactoryImpl -Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis.soap.SOAPConnectionFactoryImpl -Djavax.xml.soap.SOAPFactory=org.apache.axis.soap.SOAPFactoryImpl -Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactoryImpl"

There are a number of ways to set JVM options. For example, you can add your JAVA_OPTS settings to these files:

File	**Add JVM Options Here**
<tomcat>/bin/setclasspath.bat	`set JAVA_ENDORSED_DIRS=%BASEDIR%\common\endorsed`
<tomcat>/bin/setclasspath.sh	`JAVA_ENDORSED_DIRS="$BASEDIR"/common/endorsed`
<tomcat>/bin/setenv.bat or <tomcat>/bin/setenv.sh	JAVA_OPTS setting can go anywhere in this file.
<jboss>/bin/run.bat <jboss>/bin/run.sh	`set JAVA_OPTS=%JAVA_OPTS% -Dprogram.name=%PROGNAME%` or `export JAVA_OPTS="$JAVA_OPTS -Dprogram.name=$PROGNAME"`

Appendix C Troubleshooting

C.1 Build Troubleshooting

C.1.1 Name Undefined Error (Old Ant Version)

If you are not using the bundled version of Apache Ant included with the JasperReports Server source code package, you could get the following error when running the buildomatic scripts:

```
BUILD FAILED
c:\js-builds\jasperserver\buildomatic\install.xml:6: Problem: failed to create task or
type if
Cause: The name is undefined.
 Action: Check the spelling.
 Action: Check that any custom tasks/types have been declared.
 Action: Check that any <presetdef>/<macrodef> declarations have taken place.
```

Solution:

The buildomatic scripts require ant version 1.8.1 or higher. Additionally, the ant-contrib.jar file needs to be included in your ant/lib directory. If you are running with your own ant version, you can copy this jar to your <ant-home>/lib directory:

From:

<js-src>/apache-ant/lib/ant-contrib.jar   or
<js-src>/jasperserver/buildomatic/extra-jars/ant-contrib.jar

To:

<ant-home>/lib	(General example)
C:\apache-ant-1.8.1\lib	(Windows example)
/usr/share/java/apache-ant/lib	(Linux example)
/usr/share/ant/lib	(Mac example)

C.1.2 Linux/Mac Error with Maven /usr/boot

When building under Linux, it is possible to get an error similar to the following:

```
BUILD FAILED
/home/devuser/js-builds/jasperserver/buildomatic/bin/dev.xml:91:
/usr/boot does not exist
```

Buildomatic attempts to find the MAVEN_HOME setting and can be unsuccessful when the maven binary is installed in the /usr/bin/mvn location. A quick workaround is to update you default_master.properties file:

cd <js-src>/jasperserver/buildomatic

edit default_master.properties

Look for the line:

maven = /usr/bin/mvn

Below this line, add the line:

maven.home = /usr/share/maven2

This same issue can be seen when executing Maven on a Mac:

cd <js-src>/jasperserver/buildomatic

edit default_master.properties

Look for the line:

maven = /usr/bin/mvn

Below this line, add the line:

maven.home = /usr/share/maven

C.2 Database Troubleshooting

The most common error encountered when building JasperReports Server involves the database. These errors often result from not being able to connect to the database. For information about database connection problems, see the Troubleshooting Appendix of the *JasperReports Server Installation Guide*.

C.3 Maven Troubleshooting

C.3.1 Maven Binary Versions

The recommended Maven version is 3.0.4. The source build also works with Maven 2.2.1. Jaspersoft has found that Maven version 3.0.3 can get errors resolving dependencies; therefore this version should not be used.

C.3.2 Clear JasperReports Server Artifacts in Maven Local Repository

If you have an existing source build environment and you add new code, such as a bug fix source patch update, you can clear the JasperReports Server artifacts in your Maven local repository to ensure that the newly built artifacts contain the necessary new content. Maven updates the artifacts automatically, but if you havetrouble building or pulling in the modified code, you can try deleting these artifact trees.

To clear existing JasperReports Server artifacts:

1. Go to the repository directory:

    ```
    cd <home-dir-path>/.m2/repository
    ```

2. Remove the old versions by deleting the following directories and their contents:

`com/jaspersoft`:	Community Project artifact tree
`jaspersoft`:	Commercial version artifact tree

C.3.3 Clear Entire Local Repository

If you want to completely rebuild everything, remove all of the cached jars in your Maven local repository. To do this you can delete (or rename) the entire local repository.

Then when you build JasperServer, all dependencies are re-downloaded.

`cd <home-dir-path>/.m2`

remove folder: `repository`

C.3.4 Maven Warnings

Maven2 generates warnings during the artifact validation process. Warnings regarding non-standard layouts of artifacts, such as a JAR file not having a corresponding POM file and a checksum file being unavailable, are common and can typically be ignored.

The following example shows a warning, even though the required JAR file was downloaded successfully:

```
[WARNING] Unable to get resource from repository jasperServer (file://C:/svn/js-
buildlds/jasperserver-repo
Downloading: http://repo1.maven.org/maven2/commons-logging/commons-logging/1.0/commons-
logging-1.0.pom
163b downloaded
```

C.3.5 Maven Error: Transferring Files

With the Maven build, there are many files that are downloaded on the very first build. It is not unusual to get an error downloading a file. You can usually get around a file transfer error by kicking off the build again.

In the following example, there was a transfer error on the castor.jar file:

```
[ERROR] BUILD ERROR
[INFO] ------------------------------------------------------------------------
[INFO] Error building POM (may not be this project's POM).
Project ID: castor:castor
Reason: Error getting POM for 'castor:castor' from the repository: Error transferring
file
castor:castor:pom:1.0

from the specified remote repositories:
Maven Snapshots (http://snapshots.maven.codehaus.org/maven2/),
central (http://repo1.maven.org/maven2),
ApacheSVN-central (http://svn.apache.org/maven-snapshot-repository),
jasperServer (file://C:\jasperserver-3.7.0-src\jasperserver-repo\jasperserver-maven)
```

Such problems can be fixed by re-running the `mvn install` command, which effectively restarts the build.

C.3.6 Maven Build Error: Failed to Resolve Artifact

In some cases, Maven may return the following error:

```
[ERROR] BUILD ERROR
[INFO] ------------------------------------------------------------------------
[INFO] Failed to resolve artifact.
Missing:
----------
1) javax.transaction:jta:jar:1.0.1B

  Try downloading the file manually from:
  http://java.sun.com/products/jta

Then, install it using the command:
  mvn install:install-file -DgroupId=javax.transaction -DartifactId=jta \
    -Dversion=1.0.1B -Dpackaging=jar -Dfile=/path/to/file
Path to dependency:
  1) com.jaspersoft.jasperserver.api.metadata:jasperserver-api-metadata:jar:3.0.0
  2) org.springframework.security:spring-security:jar:2.0-m2
  3) org.springframework:spring-jdbc:jar:2.0-m2
  4) org.springframework:spring-dao:jar:2.0-m2
  5) javax.transaction:jta:jar:1.0.1B


2) jasperreports:jasperreports:jar:3.0.0

  Try downloading the file manually from the project website.
    mvn install:install-file -DgroupId=jasperreports -DartifactId=jasperreports \
    -Dversion=3.0.0 -Dpackaging=jar -Dfile=/path/to/file
Path to dependency:
  1) com.jaspersoft.jasperserver.api.metadata:jasperserver-api-metadata:jar:3.1.0
  2) jasperreports:jasperreports:jar:3.1.0
----------
2 required artifacts are missing.


for artifact:
  com.jaspersoft.jasperserver.api.metadata:jasperserver-api-metadata:jar:3.1.0
from the specified remote repositories:
  Maven Snapshots (http://snapshots.maven.codehaus.org/maven2/),
  central (http://repo1.maven.org/maven2),
  ApacheSVN-central (http://svn.apache.org/maven-snapshot-repository),
  jasperServer (file://C:\jasperserver-3.7.0-src\jasperserver-repo)
```

This error may indicate that the setting.xml file doesn't point correctly to the jasperserver-repo directory.

In this case, many of the dependent JARs cannot be found. To resolve the problem, double-check the $HOME/.m2/settings.xml file and ensure that it properly specifies the <js-src>/jasperserver-repo directory. See section **5.3, "Creating the settings.xml File," on page 18**.

C.4 Other Build Troubleshooting

C.4.1 Error When Building Database Scripts

You may get an error when compiling in the jasperserver-repository-hibernate/build-db directory with the following message:

```
[ERROR] BUILD ERROR
[INFO] ------------------------------------------------------------------------
[INFO] Error executing ant tasks
Embedded error: Source file does not exist!
```

The most likely problem is that your .m2/settings.xml file doesn't point to the correct source location, and the build step did not find the Quartz scripts. The settings.xml file should contain the path to the quartz script corresponding to your database, for example:

<js.quartz.script>/home/<user>/<js-src>/jasperserver/scripts/quartz/tables_<database>.sql</js.quartz.script>

If you use the buildomatic scripts you should not get this kind of error.

See section **5.3, "Creating the settings.xml File," on page 18**.

C.4.2 Error Message to Ignore

Database error messages are also common when running in the build-db directory, because the generated scripts, which are the output of the build step, attempt to clean up any database tables that already exist. If the tables do not exist, an error message that looks like this is returned:

```
[hibernatetool] Error #1: java.sql.SQLException: Table 'jasperserver.jiuserrole'
doesn't exist
```

Such errors can be safely ignored.

C.5 PostgreSQL 8.1 Error on Sugarcrm DB Load

If you are using an older version of PostgreSQL, version 8.1 or earlier, you get an error when loading the sugarcrm sample database. This database needs to exist in order for the integration-tests to execute cleanly.

The error seen would be similar to the following:

Failed to execute: ALTER SEQUENCE bugs_bug_number_seq OWNED BY bugs.bug_number

org.postgresql.util.PSQLException: ERROR: syntax error at or near "OWNED"

This due to a change in the SQL statement syntax after PostgreSQL 8.1.

To get around this issue, you can comment out the ALTER statements found in the file:

<js-src>/jasperserver/buildomatic/install_resources/sql/postgresql/sugarcrm.zip

The commented out line would look like this (two dashes mark line as a comment):

-- ALTER SEQUENCE bugs_bug_number_seq OWNED BY bugs.bug_number;

To make this change, you should unzip the file, alter the sugarcrm.sql file, then re-zip the file. Delete the sugarcrm.sql file because the buildomatic step that executes the database load uses the sugarcrm.zip file.