



JASPERREPORTS SERVER COMMUNITY PROJECT SOURCE BUILD GUIDE

RELEASE 4.0

<http://jasperforge.org>

Copyright © 2011 JasperSoft Corporation. All rights reserved. Printed in the U.S.A. JasperSoft, the JasperSoft logo, JasperAnalysis, JasperServer, JasperETL, JasperReports, JasperReports Server, and iReport, are trademarks and/or registered trademarks of JasperSoft Corporation in the United States and in jurisdictions throughout the world. All other company and product names are or may be trade names or trademarks of their respective owners.

This is version 0211-JSO40-4 of the *JasperReports Server CP Source Build Guide*.

TABLE OF CONTENTS

Chapter 1	Introduction	7
1.1	Supported Build Configurations	7
1.2	JasperReports Server Source Code Archives	8
Chapter 2	Components Required for Source Build	9
2.1	Checking Your Java JDK	9
2.2	Installing Maven	9
2.3	Checking Your Application Server	10
2.4	Checking Your Database Instance	10
Chapter 3	Building JasperReports Server Source Code	11
3.1	Introduction to Buildomatic Source Build Scripts	11
3.2	Downloading and Unpacking JasperReports Server Source Code	11
3.2.1	Downloading the Source Archive	11
3.2.2	Unpacking the Source Archive	12
3.2.3	Source Code Package Structure	12
3.3	Checking Apache Ant	12
3.3.1	Using Your Own Apache Ant: Get ant-contrib.jar	12
3.4	Configuring the Buildomatic Property File	12
3.4.1	Configuring the default_master.properties File	12
3.4.2	Refreshing your Buildomatic Settings	14
3.5	Building JasperReports Server	14
3.5.1	Running Ant in Debug Mode	15
3.6	Setting Java JVM Options	15
3.7	Starting Your Application Server	15
3.8	Logging into JasperReports Server	15
3.9	JasperReports Server Log Files	15
Chapter 4	Additional Buildomatic Information	17
4.1	Generated Property Files	17

4.2	Existing and Generated Database SQL Files	17
4.3	Generated WAR File Location and deploy-webapp-ce Target	18
Chapter 5	Configuring the Build Environment Manually	19
5.1	Setting Up the JasperReports Server Build Manually	19
5.2	Setting the Maven Java Memory Option	20
5.3	Creating the settings.xml File	20
5.3.1	Additional Step for DB2	20
5.4	Creating the hibernate.cfg.xml File	21
5.5	Creating the js.jdbc.properties File	21
5.5.1	Additional Step for Oracle	22
5.5.2	Additional Step for DB2	22
5.6	Creating the js.quartz.properties File	22
5.7	Setting Up the JDBC Driver	22
5.8	Creating JasperReports Server Databases	22
5.8.1	Creating Databases: MySQL	23
5.8.2	Additional Databases	23
5.9	Building the JasperReports Server Source Code Manually	23
5.10	Copying the JDBC Driver JAR	24
5.11	Validating Tomcat Related Configuration Files	24
5.11.1	Validating Context.xml	24
5.11.2	Validating Other Database Related Files	25
5.12	Copying the JasperReports Server WAR File to Tomcat	25
5.13	Starting JasperReports Server and Logging In	25
Chapter 6	Using Maven Repository Archive Package (3rd Party JARs)	27
6.1	Download the Maven Repository ZIP File	27
6.2	Unpacking the Maven Repository ZIP File	27
Appendix A	Java JVM Settings	29
A.1	Java 1.5	29
A.1.1	General Java 1.5 JVM Settings	29
A.1.2	Additional Java 1.5 JVM Setting for Oracle Localization	29
A.2	Java 1.6	30
A.2.1	General Java 1.6 JVM Settings	30
A.2.2	Additional Java 1.6 JVM Settings for Web Services	30
Appendix B	Troubleshooting	33
B.1	Build Troubleshooting	33
B.1.1	Name Undefined Error	33
B.2	Database Troubleshooting	33
B.3	Maven Troubleshooting	34
B.3.1	Maven Binary Versions	34
B.3.2	Clear JasperReports Server Artifacts in Maven Local Repository	34
B.3.3	Maven Warnings	34

B.3.4	Maven Error: Transferring Files	34
B.3.5	Maven Build Error: Failed to Resolve Artifact	35
B.4	Other Build Troubleshooting	36
B.4.1	Error When Building Database Scripts	36
B.4.2	Error Message to Ignore	36

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.

If you want to extend your knowledge of Jaspersoft BI software, our Ultimate Guides document its 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](#). Professional and Enterprise customers can download them from the [support portal](#).

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, JBoss or GlassFish	MySQL
	PostgreSQL

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-4.0.0-src.zip	JasperReports Server source code	Chapter 2 Chapter 3
jasperreports-server-cp-4.0.0-maven-repository.zip	Optional archive of 3rd-party dependencies	Chapter 6

The location where you unpack these archive files is referred to as <js-src> in this document.

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 1.5 or Java 1.6. JasperReports Server will not run with Java 1.4. In order to support Java 1.5, JasperReports Server source code does not contain any Java 1.6-specific language syntax.

To check your 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://java.sun.com/javase/downloads/index.jsp>.

JasperReports Server can run with some other JDK implementations such as OpenJDK 6. However, the source build procedure has been specifically tested with the Sun JDK.

2.2 Installing Maven

Apache Maven is used to compile, build, and package the JasperReports Server source code.

The JasperReports Server development team uses Maven2 because of its ability to manage third party tool dependencies via remote, online repositories. Third party tools are typically packaged as Java archive files (JARs). Information on Maven2 can be found on the Maven website: <http://maven.apache.org>

You can download and install Maven2 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:

```
mvn -version
```

The JasperReports Server source code has been test built with Maven version 2.2.1. The source build works with Maven 2.2.1 and 3.0.2. For information on supported Maven versions see section **B.3, “Maven Troubleshooting,” on page 34**

2.3 Checking Your Application Server

To run JasperReports Server, you will need an application server installed. The application server must be stopped during the build and installation procedures, except for Glassfish, which must be running.

The application servers supported by this build procedure are listed in section [1.1, “Supported Build Configurations,” on page 7](#). JasperReports Server can also run with additional application servers. For information on all supported application servers, refer to the *JasperReports Server Installation Guide*.

2.4 Checking Your Database Instance

To run JasperReports Server, you will need a database instance. The database server must be running during the build and installation procedures.

The databases supported by this build procedure are listed in section [1.1, “Supported Build Configurations,” on page 7](#).

CHAPTER 3 BUILDING JASPERREPORTS SERVER SOURCE CODE

The following sections give the 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 configuration, build, and deployment for the JasperReports Server source code.

For your convenience, Apache Ant is also 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

The JasperReports Server Community code can be downloaded as a zip file from the JasperForge.org site.

Also, look in the Source and Wiki areas for information on doing a Subversion source control checkout of the jasperserver public source code.

Download source zip package from the Release menu area of JasperForge.org:

<http://jasperforge.org/projects/jasperserver>

The source code package is a ZIP archive named:

jasperreports-server-cp-4.0.0-src.zip.

3.2.2 Unpacking the Source Archive

Unpack the `jasperreports-server-cp-4.0.0-src.zip` file to a directory location, such as `C:\` or `/home/<user>`. The resulting location will be referred to as `<js-src>` in this document:

Windows: `<js-src>` example is `C:\jasperreports-server-cp-4.0.0-src`

Linux: `<js-src>` example is `/home/<user>/jasperreports-server-cp-4.0.0-src`

3.2.3 Source Code Package Structure

After unpacking, the source directory has the following structure:

Directory	Description
<code><js-src>/apache-ant</code>	Bundled version of Apache Ant
<code><js-src>/jasperserver</code>	JasperReports Server open source code for general features
<code><js-src>/jasperserver-repo</code>	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. Shell scripts for Windows and Linux are included that are pre-configured to use the bundled version of Apache Ant. These scripts are called from the command line with the following command:

```
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, you will need to 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.

3.4 Configuring the Buildomatic Property File

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.



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.

3.4.1 Configuring 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, to which you must add your specific settings.

To create your `default_master.properties` file, follow the steps in the section below that corresponds to your database type.

3.4.1.1 MySQL

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\mysql_master.properties default_master.properties`

Linux: `cp sample_conf/source/mysql_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 = tomcat6 (or tomcat5 or jboss or glassfish2)
appServerDir	appServerDir = C:\\apache-tomcat-6.0.26 appServerDir = /home/user/apache-tomcat-6.0.26
dbHost	dbHost = localhost
dbUsername	dbUsername = root
dbPassword	dbPassword = password
maven	maven = C:\\apache-maven-2.2.1\\bin\\mvn maven = /usr/bin/mvn
js-path	js-path = C:\\jasperreports-server-cp-4.0.0-src\\jasperserver js-path = /home/<user>/jasperreports-server-cp-4.0.0-src/jasperserver
js-repo-path	js-repo-path = C:\\jasperreports-server-4.0.0-src\\jasperserver-repo js-repo-path = /home/<user>/jasperreports-server-cp-4.0.0-src/jasperserver-repo

3.4.1.2 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 = tomcat6 (or tomcat5 or jboss or glassfish2)
appServerDir	appServerDir = C:\\apache-tomcat-6.0.26 appServerDir = /home/user/apache-tomcat-6.0.26
dbHost	dbHost=localhost
dbUsername	dbUsername=postgres
dbPassword	dbPassword=postgres
maven	maven = C:\\apache-maven-2.2.1\\bin\\mvn maven = /usr/bin/mvn
js-path	js-path = C:\\jasperreports-server-cp-4.0.0-src\\jasperserver js-path = /home/<user>/jasperreports-server-cp-4.0.0-src/jasperserver
js-repo-path	js-repo-path = C:\\jasperreports-server-cp-4.0.0-src\\jasperserver-repo js-repo-path = /home/<user>/jasperreports-server-cp-4.0.0-src/jasperserver-repo

3.4.2 Refreshing your Buildomatic Settings

If you later make a change to your master properties file, you can explicitly clean and regenerate your buildomatic script settings by running Ant with the following targets:

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



Additionally, anytime the `default_master.properties` is modified, your configuration settings will automatically get re-generated (i.e. into the `buildomatic/build_conf` directory).

3.5 Building JasperReports Server

Now that the `default_master.properties` file has been edited, you can now start building the JasperReports Server source code. Make sure that your database server is running and that your application server is stopped (unless glassfish which should be running).

Once you execute the first build target, the buildomatic scripts will 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, you should look for the message `BUILD SUCCESSFUL`.

Execute the following steps at the command line (under linux, if `js-ant` does not work type: `./js-ant`):

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



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

- Delete any existing jasperserver WAR file.
- Copy the JDBC driver to the appropriate application server directory.
- Add a data source definition to the appropriate application server directory.
- Deploy the newly built jasperserver WAR file.
- Delete within the application server work directory (to clear out compiled JSP files and other cached files).

3.5.1 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 Setting Java JVM Options

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 A, “Java JVM Settings,” on page 29](#).

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 in with credentials for the JasperReports Server system administrators:

- ♦ Default admin
 - User ID: `jasperadmin`
 - Password: `jasperadmin`
- ♦ Sample User
 - User ID: `joeuser`
 - Password: `joeuser`

Once you are logged into JasperReports Server, you can create reports, run reports, create dashboards, create Domains, etc. Refer to the *JasperReports Server User Guide* for more information on the application.

If you are unable to login or have other problems, refer to [Appendix B, “Troubleshooting,” on page 33](#), 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 location (using Tomcat as an 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 that allow for the setup and configuration of a number of databases and application servers. The following sections give the locations of some of these files.

4.1 Generated Property Files

After you set your database and application server property values, you initiate buildomatic which automatically generates the database and application server configuration files needed to run JasperReports Server. You will find the generated property files in the following location:

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

Here are some of the key configuration files:

```
js.jdbc.properties
js.quartz.properties
js-glassfish-ds.xml
js-jboss-ds.xml
maven_settings.xml
```

More generated property files are located in:

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

In this directory you will find config files such as:

```
META-INF/context.xml
WEB-INF/hibernate.properties
WEB-INF/js.quartz.properties
```

The autogenerated files above are removed if you run the buildomatic target: `clean-config`. You can then regenerate them by running the target: `gen-config`. (Also, after running `clean-config`, any subsequent target will regenerate the configuration files.)

4.2 Existing and Generated Database SQL Files

Buildomatic comes with files that support a number of databases. They are located in:

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

The following files are already included in this directory to support customer installation:

js-create.ddl
js-drop.ddl
quartz.ddl
etc...

The following database files needed to create the JasperReports Server repository database schema are automatically generated by the JasperReports Server source code build:

js-create.ddl
js-drop.ddl

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

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

Then, these 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 buildomatic finishes creating the jasperserver WAR file, it does a copy to a convenient location inside of the buildomatic directory tree for use by subsequent buildomatic targets. This all occurs when the `build-ce` target is run.

This location is the following:

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

When you later run the buildomatic target `deploy-webapp-ce` here are some of the actions which take place (using Tomcat as an 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/mysql/jdbc/mysql-connector-java-5.1.10.jar`

Copied to: `<tomcat>/lib`

CHAPTER 5 CONFIGURING THE BUILD ENVIRONMENT MANUALLY

The buildomatic scripts used in the previous chapter set up all of the build configuration files needed to compile and deploy JasperReports Server.

This section describes the “old” style manual configuration to build the JasperReports Server source.



The recommended way to build the JasperReports Server source code is to use the buildomatic scripts as described in chapter [Chapter 3, “Building JasperReports Server Source Code,”](#) on page 11.

These sample manual steps mainly specify the settings for a Tomcat application server and a MySQL database. Other application servers and database combinations follow a similar configuration pattern.

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 11.

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 under 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: cd "%Documents and Settings\<user>" (Win XP)
          mkdir .m2

Linux:    cd $HOME
          mkdir .m2
```

Note the period (.) in `.m2` that indicates it is a hidden directory.

5.2 Setting the Maven Java Memory Option

When you run the JasperReports Server unit-tests, they can fail with an out of memory error. In order to have Maven set a larger default 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: copy <js-src>\jasperserver\scripts\dev-setup\settings.xml "C:\Documents and
Settings\<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 for you automatically. For more information, see chapter [Chapter 3, “Building JasperReports Server Source Code,” on page 11](#).

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-4.0.0-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-4.0.0-src/jasperserver-repo</url>
</repository>
```

5.3.1 Additional Step for DB2

For DB2, uncomment the following values in the sample settings.xml file to set the Quartz delegate and the schema prefix:

```
<!-- DB2 specific setting for Quartz. Ends up in WEB-INF/js.quartz.properties -->
<quartz.delegate>org.quartz.impl.jdbcjobstore.DB2v8Delegate</quartz.delegate>

<!-- DB2 (or Oracle) specific setting for Quartz to handle schema name. -->
<!-- Ends up in WEB-INF/js.quartz.properties -->
<quartz.tablePrefix>JSPRSRVR.QRTZ_</quartz.tablePrefix>
```

Also, make sure your Quartz script setting points to the script that is specific to DB2:

```
<js.quartz.script>/home/<user>/jasperreports-server-4.0.0-src/jasperserver/scripts/
quartz/tables_db2.sql</js.quartz.script>
```

5.4 Creating the hibernate.cfg.xml File

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

Windows: copy <js-src>\jasperserver\scripts\dev-setup\hibernate.cfg.xml
"C:\Documents and Settings\<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">com.mysql.jdbc.Driver</property>
<property name="connection.url">jdbc:mysql://localhost:3306/jasperserver?useUnicode=
  true&characterEncoding=UTF-8</property>
<property name="connection.username">jasperdb</property>
<property name="connection.password">password</property>

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

5.5 Creating the js.jdbc.properties File

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

Windows: copy <js-src>\jasperserver\scripts\dev-setup\js.jdbc.properties
"C:\Documents and Settings\<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.MySQLInnoDBDialect

metadata.jdbc.driverClassName=com.mysql.jdbc.Driver
metadata.jdbc.url=jdbc:mysql://localhost:3306/jasperserver?useUnicode=true&
  characterEncoding=UTF-8
metadata.jdbc.username=jasperdb
metadata.jdbc.password=password
metadata.jdbc.database=jasperserver

metadata.jndi=jdbc/jasperserver

test.jdbc.driverClassName=com.mysql.jdbc.Driver
test.jdbc.url=jdbc:mysql://localhost:3306/sugarcrm
test.jdbc.username=jasperdb
test.jdbc.password=password

test.jndi=jdbc/sugarcrm

foodmart.jdbc.driverClassName=com.mysql.jdbc.Driver
foodmart.jdbc.url=jdbc:mysql://localhost:3306/foodmart
foodmart.jdbc.username=jasperdb
foodmart.jdbc.password=password

foodmart.jndi=jdbc/foodmart
```

5.5.1 Additional Step for Oracle

In order for the unit-tests to pass under Oracle, you must uncomment the `upperCaseNames` properties to enable them:

```
test.databaseFlavor=oracle
test.foodmart.upperCaseNames=true
test.sugarcrm.upperCaseNames=true
```

5.5.2 Additional Step for DB2

In order for the unit-tests to pass under DB2, you must uncomment the `upperCaseNames` properties in order to enable them:

```
test.databaseFlavor=db2
test.foodmart.upperCaseNames=true
test.sugarcrm.upperCaseNames=true
```

5.6 Creating the `js.quartz.properties` File

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

```
Windows: copy <js-src>\jasperserver\scripts\dev-setup\js.quartz.properties
         "C:\Documents and Settings\<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
```

5.7 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 unit-tests.

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

```
<js-src>/jasperserver-repo/mysql/mysql-connector/java-3.1.11-bin/mysql-connector-java-3.1.11-bin.jar
```

5.8 Creating JasperReports Server Databases

JasperReports Server runs with a repository database that is typically named `jasperserver`. Additionally, in order to run the unit-tests, you must create the sample databases `sugarcrm` and `foodmart`.

5.8.1 Creating Databases: MySQL

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

Parameter	Default Value
Database Host Name	localhost
Database Port	3306
Database User Name	jasperdb
Database User Name (alternate)	root
Database Password	password

If your MySQL instance and database use the values above, you only need to make minimal changes to the JasperReports Server build configuration files. If you prefer to not use the root MySQL account for development, you can create a database user with the name of `jasperdb` for JasperReports Server to use.

The following example commands create a new database user:

```
mysql -u root -p
mysql> grant all on *.* to jasperdb@localhost identified by 'password';
mysql> flush privileges;
```

If you are going to access MySQL on a remote server, run an additional grant statement:

```
mysql> grant all on *.* to jasperdb@'%' identified by 'password';
```

If you use different values, specify them in the build configuration files described in [5.1, “Setting Up the JasperReports Server Build Manually,”](#) on page 19.

Now log into MySQL and create the databases:

```
cd <js-src>/jasperserver/buildomatic/install_resources/sql/mysql
mysql -u jasperdb -p -h localhost
mysql> create database jasperserver character set utf8;
mysql> create database sugarcrm;
mysql> create database foodmart;
mysql> use sugarcrm;
mysql> source sugarcrm-mysql.sql;
mysql> use foodmart;
mysql> source foodmart-mysql.sql;
```

5.8.2 Additional Databases

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

5.9 Building the JasperReports Server Source Code Manually

To build the JasperReports Server Professional and Enterprise source code, you must first build the open source JasperReports Server Community Edition code.

Maven will download most of the 3rd party jar dependencies from external, remote repositories. In order to skip the download of these dependencies, you can use the contents of the `jasperserver-4.0.0-maven-repository.zip` file. For information on installing this archive see chapter [Chapter 6, “Using Maven Repository Archive Package \(3rd Party JARs\),”](#) on page 27.

```
cd <js-src>/jasperserver
mvn clean install
cd <js-src>/jasperserver/jasperserver-repository-hibernate/build-db
mvn clean install
cd <js-src>/jasperserver/jasperserver-unit-test
mvn clean install
```

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

If you encounter errors, refer to section [B.1, “Build Troubleshooting,” on page 33](#) for help with debugging.

5.10 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/mysql/jdbc/mysql-connector-java-5.1.10.jar` to the following location:

Tomcat 5.5: `<tomcat>/common/lib`

Tomcat 6.0: `<tomcat>/lib`

5.11 Validating Tomcat Related Configuration Files

5.11.1 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 MySQL 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="jasperdb" password="password" driverClassName="com.mysql.jdbc.Driver"
validationQuery="select 1" testOnBorrow="true"
url="jdbc:mysql://localhost:3306/jasperserver?useUnicode=true&
characterEncoding=UTF-8&autoReconnect=true&autoReconnectForPools=true"/>

<Resource name="jdbc/sugarcrm" auth="Container" type="javax.sql.DataSource"
maxActive="100" maxIdle="30" maxWait="10000"
username="jasperdb" password="password" driverClassName="com.mysql.jdbc.Driver"
validationQuery="select 1" testOnBorrow="true"
url="jdbc:mysql://localhost:3306/sugarcrm?useUnicode=true&
characterEncoding=UTF-8&autoReconnect=true&autoReconnectForPools=true"/>

<Resource name="${foodmart.jndi}" auth="Container" type="javax.sql.DataSource"
maxActive="100" maxIdle="30" maxWait="10000"
username="jasperdb" password="password" driverClassName="com.mysql.jdbc.Driver"
validationQuery="select 1" testOnBorrow="true"
url="jdbc:mysql://localhost:3306/foodmart?autoReconnect=true&
autoReconnectForPools=true"/>
```



In an XML file, the ampersand (&) must be escaped in the URL line. JasperReports Server will return a database startup error if the URL doesn't have the correct form. The following is the correct format for XML:

```
UseUnicode=true&amp;characterEncoding=UTF-8
```

The following format will cause an error if used in XML:

```
useUnicode=true&characterEncoding=UTF-8
```

When such a URL is encountered, Tomcat returns the following error:

```
SAXParseException: The reference to entity "characterEncoding" must end with the ';'
delimiter
```

5.11.2 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
```

5.12 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

5.13 Starting JasperReports Server and Logging In

You can now start your application server which will start JasperReports Server. Also, first make sure that your database is running.

Enter the login URL with the default port number:

```
http://localhost:8080/jasperserver
```

Log in with credentials for the JasperReports Server system administrators:

- ♦ Default admin - User ID `jasperadmin` and password `jasperadmin`.

CHAPTER 6 USING MAVEN REPOSITORY ARCHIVE PACKAGE (3RD PARTY JARs)

When you build JasperReports Server, Maven will handle downloading all of the necessary third party JAR dependencies. By default, Maven connects to the web to locate and download any JARs it doesn't find locally. This can result in long build times, particularly if you have a slow internet connection.

To reduce the time it takes to build, and to get around any download errors, you can use the ZIP package `jasperreports-server-4.0.0-maven-repository.zip`. It includes all of the third party JAR dependencies in a Maven local repository file structure. This eliminates the need to download each individual JAR while the JasperReports Server build executes.

However, if you want to get the latest JAR files from the remote maven repositories instead of the ones provided, skip this step.

6.1 Download the Maven Repository ZIP File

Download this package from the JasperForge.org Release menu area:

<http://jasperforge.org/projects/jasperserver>

Look for a file with the following name:

`jasperreports-server-cp-4.0.0-maven-repository.zip`

6.2 Unpacking the Maven Repository ZIP File

Unpacking the Maven repository ZIP file into your Maven `<home>/ .m2` directory creates a directory called `repository`. This directory represents the Maven local repository:

Windows: `C:\Documents and Settings\<user>\.m2\repository`

Linux: `/home/<user>/.m2/repository`

APPENDIX A JAVA JVM SETTINGS

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

A.1 Java 1.5

The following tables give the recommended settings for Java 1.5 and a localization setting that applies to Oracle only. You can also copy these settings from the files in the `<js-src>/jasperserver/scripts/java-settings` directory.

A.1.1 General Java 1.5 JVM Settings

The following settings can be increased or decreased for your particular requirements.

General Java 1.5 JVM Settings on Windows	
Tomcat file	<code><tomcat>\bin\setclasspath.bat</code> or <code>setenv.bat</code>
JBoss file	<code><jboss>\bin\run.bat</code>
JAVA_OPTS setting	<pre>set JAVA_OPTS=%JAVA_OPTS% -Xms512m -Xmx1024m -XX:PermSize=32m -XX:MaxPermSize=128m -Xss2m set JAVA_OPTS=%JAVA_OPTS% -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:+CMSPermGenSweepingEnabled</pre>

General Java 1.5 JVM Settings on Linux	
Tomcat file	<code><tomcat>/bin/setclasspath.sh</code> or <code>setenv.sh</code>
JBoss file	<code><jboss>/bin/run.sh</code>
JAVA_OPTS setting	<pre>JAVA_OPTS="\$JAVA_OPTS -Xms512m -Xmx1024m -XX:PermSize=32m -XX:MaxPermSize=128m -Xss2m" JAVA_OPTS="\$JAVA_OPTS -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:+CMSPermGenSweepingEnabled"</pre>

A.1.2 Additional Java 1.5 JVM Setting for Oracle Localization

To support localization when running JasperReports Server with Oracle hosting its repository, you must set an additional Java option:

Additional Java 1.5 JVM Setting for Oracle Localization on Windows	
Tomcat file	<tomcat>\bin\setclasspath.bat or setenv.bat
JBoss file	<jboss>\bin\run.bat
JAVA_OPTS setting	set JAVA_OPTS=%JAVA_OPTS% -Doracle.jdbc.defaultNChar=true

Additional Java 1.5 JVM Setting for Oracle Localization on Linux	
Tomcat file	<tomcat>/bin/setclasspath.sh or setenv.sh
JBoss file	<jboss>/bin/run.sh
JAVA_OPTS setting	JAVA_OPTS="\$JAVA_OPTS -Doracle.jdbc.defaultNChar=true"

A.2 Java 1.6

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



The settings described apply specifically to the Sun JVM. Other JVM implementations may or may not have equivalent settings.

A.2.1 General Java 1.6 JVM Settings

The following settings can be increased or decreased for your particular requirements.

General Java 1.6 JVM Settings on Windows	
Tomcat file	<tomcat>\bin\setclasspath.bat or setenv.bat
JBoss file	<jboss>\bin\run.bat
JAVA_OPTS setting	<pre>set JAVA_OPTS=%JAVA_OPTS% -Xms512m -Xmx1024m -XX:PermSize=32m -XX:MaxPermSize=128m set JAVA_OPTS=%JAVA_OPTS% -Xss2m -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled</pre>

General Java 1.6 JVM Settings on Linux	
Tomcat file	<tomcat>/bin/setclasspath.sh or setenv.sh
JBoss file	<jboss>/bin/run.sh
JAVA_OPTS setting	<pre>JAVA_OPTS="\$JAVA_OPTS -Xms512m -Xmx1024m -XX:PermSize=32m -XX:MaxPermSize=128m" JAVA_OPTS="\$JAVA_OPTS -Xss2m -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled"</pre>

A.2.2 Additional Java 1.6 JVM Settings for Web Services

Java 1.6 includes a web services implementation that can conflict with JasperReports Server's AXIS-based web services classes. These conflicts could cause JasperReports Server web services and the resources that rely on them to fail, for example the JasperReports Server plugin for iReport, web services, and analysis XML/A connections.

In order to prevent these conflicts, add the following JVM settings. If you do not use JasperReports Server web services or resources that rely on them, you do not need the settings in this table.

Additional Java 1.6 JVM Settings for Web Services on Windows	
Tomcat file	<tomcat>\bin\setclasspath.bat or setenv.bat
JBoss file	<jboss>\bin\run.bat
JAVA_OPTS settings	<pre>set JAVA_OPTS=%JAVA_OPTS% -Djavax.xml.soap.MessageFactory= org.apache.axis.soap.MessageFactoryImpl set JAVA_OPTS=%JAVA_OPTS% -Djavax.xml.soap.SOAPConnectionFactory= org.apache.axis.soap.SOAPConnectionFactoryImpl set JAVA_OPTS=%JAVA_OPTS% -Djavax.xml.soap.SOAPFactory= org.apache.axis.soap.SOAPFactoryImpl set JAVA_OPTS=%JAVA_OPTS% -Djavax.xml.transform.TransformerFactory= org.apache.xalan.processor.TransformerFactoryImpl</pre>

Additional Java 1.6 JVM Settings for Web Services on Linux	
Tomcat file	<tomcat>/bin/setclasspath.sh or setenv.sh
JBoss file	<jboss>/bin/run.sh
JAVA_OPTS settings	<pre>JAVA_OPTS="\$JAVA_OPTS -Djavax.xml.soap.MessageFactory= org.apache.axis.soap.MessageFactoryImpl " JAVA_OPTS="\$JAVA_OPTS -Djavax.xml.soap.SOAPConnectionFactory= org.apache.axis.soap.SOAPConnectionFactoryImpl " JAVA_OPTS="\$JAVA_OPTS -Djavax.xml.soap.SOAPFactory= org.apache.axis.soap.SOAPFactoryImpl " JAVA_OPTS="\$JAVA_OPTS -Djavax.xml.transform.TransformerFactory= org.apache.xalan.processor.TransformerFactoryImpl "</pre>

APPENDIX B TROUBLESHOOTING

B.1 Build Troubleshooting

B.1.1 Name Undefined Error

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 from the following location to your <ant-home>/lib directory:

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

B.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. The Troubleshooting Appendix of the *JasperReports Server Installation Guide* has the most detail on database connection problems.

B.3 Maven Troubleshooting

B.3.1 Maven Binary Versions

The current recommended Maven version is 2.2.1. It is not recommended that versions older than 2.0.9 be used.



Jaspersoft has encountered problems when building the JasperReports Server code using Maven 2.0.6; this version of Maven **does not** properly build the source code.

B.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 all the newly built artifacts contain all the necessary new content. Maven updates the artifacts automatically, but if you are having trouble 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 all their contents:

```
com/jaspersoft: Community Edition artifact tree
```

```
jaspersoft: Professional and Enterprise Edition artifact tree
```

B.3.3 Maven Warnings

Maven2 generates warnings during the artifact validation process. Warnings regarding non-standard layouts of artifacts (such as having a JAR file without a corresponding POM file, or if a checksum file is not available) 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
```

B.3.4 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.

If you are having trouble getting a particular dependent file, you can utilize the optional repository archive package described in chapter [Chapter 6, “Using Maven Repository Archive Package \(3rd Party JARs\),” on page 27](#).

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.

B.3.5 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, you must 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 20.

B.4 Other Build Troubleshooting

B.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 correctly point to your 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 20.

B.4.2 Error Message to Ignore

Database errors messages are also common when running in the `build-db` directory, because the generated scripts (that is, the output of the build step) attempt to clean up any database tables that already exist. If the tables do not exist, then there is an error message to that effect is returned. It is similar to:

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

Such errors can be safely ignored.