



# **JASPERSERVER COMMUNITY EDITION**

## **SOURCE BUILD GUIDE**

RELEASE 3.5

[jasperforge.org](http://jasperforge.org)

---

© 2009 JasperSoft Corporation. All rights reserved. Printed in the U.S.A. JasperSoft, the JasperSoft logo, JasperAnalysis, JasperServer, JasperETL, JasperReports, JasperStudio, iReport, and Jasper4 products 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 0509-JSO35-1 of the *JasperServer Community Edition Source Build Guide*.

## TABLE OF CONTENTS

---

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Supported Build Configurations	7
1.2	About JasperServer Source Code Archives	7
<b>2</b>	<b>Required Components for Source Build</b>	<b>9</b>
2.1	Check Your Java JDK	9
2.2	Installing Maven	9
2.3	Check Your Application Server	10
2.4	Check Your Database Instance	10
<b>3</b>	<b>Building JasperServer Source Code</b>	<b>11</b>
3.1	Introduction to Buildomatic Source Build Scripts	11
3.2	Download and Unpack JasperServer Source Code	11
3.2.1	Download the Source Zip	11
3.2.2	Unpack the Source Zip	11
3.2.3	Resulting Source Structure	12
3.3	Check Apache Ant	12
3.3.1	Use Your Own Apache Ant	12
3.4	Configuring the Buildomatic Source Build Scripts	12
3.4.1	Configure js-build.properties	12
3.4.2	Configure default_master.properties	13
3.4.3	How to Refresh your Buildomatic Settings	14
3.5	Building JasperServer	14
3.5.1	Run Ant in Debug Mode	15
3.6	Setting Java JVM Options	15
3.7	Start Your Application Server	16
3.8	Logging into JasperServer	16
3.9	JasperServer Logging Location	16

<b>4</b>	<b>Manual Configuration of Build Environment</b>	<b>17</b>
4.1	Manual Setup of JasperServer Build	17
4.2	Setting Maven Java Memory Option	18
4.3	Creating the settings.xml	18
4.4	Create hibernate.cfg.xml File	19
4.5	Create js.jdbc.properties File	19
4.6	Create the js.quartz.properties File	19
4.7	Setting up the JDBC Driver	20
4.8	Create JasperServer Databases	20
4.8.1	Create Databases: MySQL	20
4.8.2	Additional Databases	21
4.9	Manually Build JasperServer Source Code	21
4.10	Preparing to Run JasperServer	22
4.10.1	Copying the JDBC Driver JAR	22
4.11	Validate Tomcat-related Configuration Files	22
4.11.1	Validating Context.xml	22
4.11.2	Validating Other Database Related Files	23
4.12	Copy JasperServer WAR File to Tomcat	23
4.13	Start JasperServer and Login	23
<b>5</b>	<b>Additional Buildomatic Information</b>	<b>25</b>
5.1	Generated Property Files	25
5.2	Existing and Generated Database Files	26
5.3	Generated WAR File Location and deploy-webapp-ce Target	26
<b>Appendix A</b>	<b>Java JVM Settings</b>	<b>27</b>
A.1	Java 1.5.	27
A.1.1	General Java 1.5 JVM Settings	27
A.2	Java 1.6.	27
A.2.1	General Java 1.6 JVM Settings	28
A.2.2	Additional Java 1.6 Web Services Settings	28
<b>Appendix B</b>	<b>Troubleshooting</b>	<b>31</b>
B.1	Build Troubleshooting	31
B.1.1	"Name Undefined" Error	31
B.2	Database Troubleshooting	31
B.3	Maven Troubleshooting	32
B.3.1	Maven Binary Versions	32
B.3.2	Clear JasperServer Artifacts in Maven Local Repository	32
B.3.3	Maven Warnings	32
B.3.4	Maven Error: Transferring JAR File	33
B.3.5	Maven Build Error: Failed to resolve artifact	33
B.4	Other Build Troubleshooting	34
B.4.1	Error When Building Database Scripts	34
B.4.2	Error Message to Ignore	35

<b>Appendix C Building From JasperForge.org Public SVN Source .....</b>	<b>37</b>
<b>Glossary .....</b>	<b>39</b>



# 1 INTRODUCTION

JasperServer builds on JasperReports as a comprehensive family of Business Intelligence (BI) products, providing robust static and interactive reporting, report server, and data analysis capabilities. Jaspersoft provides the source code for JasperServer so that developers can customize and extend the application for their specific needs. This guide assists such developers in obtaining, setting up, building, and running JasperServer from its source files.



This document describes how to build from a command line shell under Linux or Windows. It doesn't 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 JasperServer Source Code Archive

In the filename of the source code archive for JasperServer, <ver> is the version of JasperServer:

Filename	Description	Documented In
jasperserver-<ver>-src.zip	JasperServer source code	Chapters <a href="#">2</a> and <a href="#">3</a>

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





## 2 REQUIRED COMPONENTS FOR SOURCE BUILD

---

The components listed in this section are required in order to build and run JasperServer.

### 2.1 Check Your Java JDK

The JasperServer source code can be compiled under Java 1.5 or Java 1.6. JasperServer will not run with Java 1.4. In order to support Java 1.5, JasperServer source code does not contain any Java 1.6 specific language syntax.

To check you Java version, run the following:

```
java -version
```

To install Java, follow the download and installation instructions found at Sun's Java web site: <http://java.sun.com/javase/downloads/index.jsp>.

JasperServer 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 then JasperServer source code.

The JasperServer development team uses Maven2 because of its ability to manage third party tool (JAR) dependencies via remote, online repositories. 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`) on your environment `PATH` so that you can execute `mvn` from the command line. To check your maven version, run the following:

```
mvn -version
```

The JasperServer source code has been test built with Maven version 2.0.9 and 2.1.0.



For information on supported Maven versions see section [B.3, “Maven Troubleshooting,”](#) on page 32.

## 2.3 Check Your Application Server

To run JasperServer, you will need an application server installed. The application servers supported by this build procedure are listed in section [1.1, “Supported Build Configurations,” on page 7](#).

JasperServer can also run with additional application servers. For information on all supported application servers, refer to the *JasperServer Installation Guide*.

## 2.4 Check Your Database Instance

To run JasperServer, you will need a database instance. The databases supported by this build procedure are listed in section [1.1, “Supported Build Configurations,” on page 7](#).

## 3 BUILDING JASPERSERVER SOURCE CODE

---

### 3.1 Introduction to Buildomatic Source Build Scripts

The JasperServer 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 JasperServer source code. For your convenience, Apache Ant is also bundled into the source code distribution to simplify the setup.

### 3.2 Downloading and Unpacking JasperServer Source Code

#### 3.2.1 Downloading the Source Zip

The JasperServer source code can be downloaded from the JasperForge.org site. The site is located here:

<http://jasperforge.org>

Navigate to the JasperServer Downloads area. The source package name is the following:

`jasperserver-<ver>-src.zip`

#### 3.2.2 Unpacking the Source Zip

Unpack the `jasperserver-<ver>-src.zip` to a directory location, such as, for instance, `C:\` or `/home/<user>`. The resulting location will be referred to as `<js-src>` in this document:

Windows: `<js-src>` example is `C:\jasperserver-<ver>-src`

Linux: `<js-src>` example is `/home/<user>/jasperserver-<ver>-src`

### 3.2.3 Source Code Package Structure

After unpacking, the source directory has the following structure:

Directory	Description
<js-src>/apache-ant	Pre-integrated (bundled) version of Apache Ant
<js-src>/jasperserver	JasperServer source code
<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. 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:

```
js-ant <target-name>
```

### 3.3.1 Using Your Own Apache Ant

Alternatively, if you prefer to use a version of Ant you have previously installed, you must do some setup steps.

1. Make sure you are using Apache Ant 1.7.0 or higher.
2. Copy the file ant-contrib-<ver>.jar from the <js-src>/apache-ant/lib directory to your <ant-home>/lib directory.

The ant-contrib-<ver>.jar enables conditional logic in Ant scripts.

## 3.4 Configuring the Buildomatic Source Build Scripts

The buildomatic scripts are found at the following location:

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

The scripts in the above directory are used to build for all supported applications.

### 3.4.1 Configuring js-build.properties

The js-build.properties file specifies the location of your Maven binary and the paths to your source code.

To configure your js-build.properties:

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

```
cd <js-src>/jasperserver/buildomatic
```
2. Edit the file js-build.properties.
3. Set the following properties according to your local settings:

Property	Description
maven	The location of your Maven executable (mvn.bat or mvn)
js-base-path	The location of the <js-src>/jasperserver directory
js-repo-path	The location of the <js-src>/jasperserver-repo directory

For example:

Property	Windows Example Linux Example
maven	C:\bin\apache-maven-2.0.9\bin\mvn.bat /usr/local/bin/mvn
js-base-path	C:\jasperserver-<ver>-src\jasperserver /home/<user>/jasperserver-<ver>-src/jasperserver
js-repo-path	C:\jasperserver-<ver>-src\jasperserver-repo /home/<user>/jasperserver-<ver>-src/jasperserver-repo



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

### 3.4.2 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.

#### 3.4.2.1 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\mysql_master.properties default_master.properties`

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

3. Edit the new default\_master.properties file and set the following properties to your local settings:

Property	Example
appServerDir	appServerDir=C:\apache-tomcat-6.0.18
dbUsername	dbUsername=root
dbPassword	dbPassword=password
dbHost	dbHost=localhost

#### 3.4.2.2 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/postgresql_master.properties default_master.properties`

Linux: `cp sample_conf/postgresql_master.properties ./default_master.properties`

3. Edit the new default\_master.properties file and set the following properties to your local settings:

Property	Example
appServerDir	appServerDir=C:\\apache-tomcat-6.0.18
dbUsername	dbUsername=postgres
dbPassword	dbPassword=postgres
dbHost	dbHost=localhost

### 3.4.3 Refreshing your Buildomatic Settings

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

```
js-ant clean-config
js-ant gen-config
```

The first target will clear the buildomatic/build\_conf/default directory. The second will re-build the configuration settings.



After the `clean-config` target has been run, any subsequent target will automatically re-build the configuration settings; `gen-config` is simply a convenience.

## 3.5 Building JasperServer

Now that the default\_master.properties file has been edited, you can now start building the JasperServer source code.

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 - (generated config files location)

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

Execute the following steps at the command line:

Commands	Description
<code>cd &lt;js-src&gt;/jasperserver/buildomatic</code>	
<code>js-ant add-jdbc-driver</code>	Installs JDBC driver to mvn local repository
<code>js-ant build-ce</code>	Builds the Community Edition source code
<code>js-ant create-js-db</code>	If the jasperserver database already exists, first run <code>js-ant drop-js-db</code>
<code>js-ant create-sugarcrm-db</code>	Creates part of the sample data
<code>js-ant load-sugarcrm-db</code>	
<code>js-ant create-foodmart-db</code>	Creates part of the sample data
<code>js-ant load-foodmart-db</code>	Runs for 10 minutes or more
<code>js-ant update-foodmart-db</code>	
<code>js-ant build-js-ddl-ce</code>	Creates the database schema files for your database type Note: there is currently a non-fatal error that can be ignored

Commands	Description
<code>js-ant init-js-db-ce</code>	Loads the schema into the database
<code>js-ant run-unit-test-ce</code>	Runs the Community Edition unit tests
<code>js-ant deploy-webapp-ce</code>	deploy to the application server



The `deploy-webapp-ce` target attempts these actions in the application server environment:

- Delete any existing jasperserver war file.
- Copy the JDBC driver to the appropriate directory (handles Tomcat 5.5, Tomcat 6, JBoss, and GlassFish).
- Add a data source definition.
- Deploy the newly built jasperserver war file.
- Delete the application server work directory (to clear out compiled JSP 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 JasperServer 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 27](#).

## 3.7 Starting Your Application Server

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

## 3.8 Logging into JasperServer

You can now login to JasperServer Professional through a web browser. If you specified all the default values when setting up JasperServer, use the following values to log in:

Login URL with the default port number:

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

Login credentials for the JasperServer administrator:

Username: jasperadmin

Password: jasperadmin

Once you are logged into the JasperServer application, you can view reports, create reports, create dashboards, create Domains, and edit resources. Refer to the *JasperServer 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 31](#)” or refer to the *JasperServer Install Guide*, which provides additional troubleshooting. For information about building other components, refer to [Appendix A, “Java JVM Settings,” on page 27](#).

## 3.9 JasperServer Log Files

The JasperServer log is written to the following location:

`<apache-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:

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



## 4 CONFIGURING THE BUILD ENVIRONMENT MANUALLY

---

The buildomatic scripts used in the previous section setup all of the build configuration files needed to compile and deploy JasperServer.

This section describes the “old” style manual configuration of the JasperServer build steps.



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

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

Begin setting up your build environment by downloading and unpacking the JasperServer source code package, as described in section 3.2, “[Downloading and Unpacking JasperServer Source Code](#),” on page 11.

### 4.1 Setting Up the JasperServer 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
```

## 4.2 Setting Maven Java Memory Option

When you run the JasperServer 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:

<code>set MAVEN_OPTS=-Xmx200m</code>	Windows
<code>export MAVEN_OPTS=-Xmx200m</code>	Linux

## 4.3 Creating the settings.xml

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

In order to configure Maven, you must create your own version of the settings.xml file. Maven expects this configuration file to be in a directory named .m2 within your home directory. To create this directory, do the following:

<code>cd \Documents and Settings\&lt;username&gt;</code>	Windows
<code>mkdir .m2</code>	
<code>cd \$HOME</code>	Linux
<code>mkdir .m2</code>	

Note the period (.) in .m2; it indicates it's a hidden directory).

The settings.xml file must reside in the root of your .m2 directory:

- C:\Documents and Settings\<username>\.m2\settings.xml
- /home/<username>/.m2/settings.xml

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



Note: If you use the buildomatic scripts to build JasperServer, all Maven settings (and other settings) are handled for you automatically. For the buildomatic script build, refer to Section 3, “[Building JasperServer Source Code](#),” on [page 11](#)

Modify items in **bold** to match your own environment.

Double-check values in settings.xml file to make sure they are correct to your environment (below has some example syntax for Linux and some for Windows):

```
<test.hibernate.cfg>/home/<username>/.m2/hibernate.cfg.xml</test.hibernate.cfg>

<test.hibernate.jdbc.properties>/home/<username>/.m2/js.jdbc.properties</test.hibernate.jdbc.properties>

<js.quartz.properties>/home/<username>/.m2/js.quartz.properties</js.quartz.properties>

<js.quartz.script>/home/<username>/js-src/jasperserver/scripts/quartz/tables_mysql_innodb.sql</js.quartz.script>

<js.quartz.script>C:/Documents and Settings/<username>/js-src/jasperserver/scripts/quartz/tables_mysql_innodb.sql</js.quartz.script>

<repository>
  <id>jasperServer</id>
  <name>Base repository for Jasper Server</name>
  <url>file:///C:/js-src/jasperserver-repo</url>
</repository>
```

## 4.4 Create hibernate.cfg.xml File

The hibernate.cfg.xml file typically resides in your Maven <home-dir>/m2 directory.



Modify items in **bold** to match your own environment.

**Table 1: Sample portion of hibernate.cfg.xml File**

```
<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>
```

## 4.5 Create js.jdbc.properties File

The js.jdbc.properties file typically resides in your Maven <home-dir>/m2 directory.



Modify items in **bold** to match your own environment.

**Table 2: Sample js.jdbc.properties File**

```
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
```

## 4.6 Create the js.quartz.properties File

The js.quartz.properties file typically resides in your Maven <home-dir>/m2 directory.

If you don't plan to connect to a mail gateway, you can use placeholder values in the `js.quartz.properties` file.



Modify items in **bold** to match your own environment.

**Table 3: Sample `js.quartz.properties` File**

```
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
```

## 4.7 Setting up the JDBC Driver

The `<home-dir>/m2/settings.xml` file specifies a JDBC driver used to generate database specific schemas as well as during the running of unit-tests.

If run JasperServer with MySQL, 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/mysql-connector-java-3.1.11.jar
(or pulls it down from the remote maven repositories)
```

**Note:** The driver listed above is fairly old. It will be updated for the next release. You can pull newer versions down from Maven repositories by specifying in your `settings.xml` file. For instance:

```
<repository.database.driver.groupId>mysql</repository.database.driver.groupId>
<repository.database.driver.artifactId>mysql-connector-java</repository.database.driver.artifactId>
<repository.database.driver.version>5.1.6</repository.database.driver.version>
```

## 4.8 Create JasperServer Databases

JasperServer 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`.

### 4.8.1 Create Databases: MySQL

The default database configuration used by the JasperServer 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 JasperServer 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 JasperServer to use.

The following example commands create a new database user:

```
mysql -u root -p                                     (log into MySQL)

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

mysql> flush privileges;                             (reload privilege
                                                         tables)
```

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';
```

Log into MySQL and create the databases:

```
cd <js-src>/scripts/mysql

mysql -u jasperdb -p -h localhost                    (Log into MySQL)

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;
```

### 4.8.2 Additional Databases

More information on manual setup of databases other than MySQL, refer to the *Additional Notes on Databases* section of the *JasperServer Install Guide*.

## 4.9 Manually Build JasperServer Source Code

Maven will download most of the 3rd party jar dependencies from external, remote repositories.

To build JasperServer, enter these commands at the command line:

```
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
```



Once you have your entire remote third party JAR dependencies downloaded, you can run the following command to recompile. This Maven offline mode saves time during compilation:

```
mvn -o clean install
or
mvn -o install
```

If you encounter errors, refer to section [B.1, “Build Troubleshooting,” on page 31](#) for help with debugging.

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 `<js-src>/jasperserver/jasperserver-war/target` directory.

## 4.10 Preparing to Run JasperServer

Before running JasperServer, you must have the Apache Tomcat application server available. This section describes application server configuration.

### 4.10.1 Copying the JDBC Driver JAR

JasperServer requires a JDBC driver to connect to its repository. You must get a JDBC driver in order to allow JasperServer (within Tomcat) to connect to the database.

Copy this file....	
<js-src>/jasperserver/scripts/drivers/mysql-connector-java-<ver>.jar	
To one of these locations.....	
Tomcat 5.5	<tomcat-install-dir>/common/lib
Tomcat 6.0	<tomcat-install-dir>/lib

## 4.11 Validate Tomcat-related Configuration Files

### 4.11.1 Validating Context.xml

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

Verify that the `context.xml` was created with the settings that you expect (that is, it matches the database settings that you put into your `$HOME/.m2/js.jdbc.properties` configuration file).

You can check this file to validate the configuration:

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



In an XML file, the ampersand (&) must be escaped in the URL line. JasperServer 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
```

#### 4.11.1.1 Context.xml

The MySQL settings are similar to the following, but 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"
  url="jdbc:mysql://localhost:3306/jasperserver?useUnicode=true&amp
  ;characterEncoding=UTF-8"/>

<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"
  url="jdbc:mysql://localhost:3306/sugarcrm"/>

<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"
  url="jdbc:mysql://localhost:3306/foodmart"/>
```

#### 4.11.2 Validating Other Database Related Files

If you have trouble running JasperServer (such as database failures) you can also check the user names, passwords, host names, and port numbers in the following files:

```
<js-src>/jasperserver/jasperserver-war/target/jasperserver/WEB-INF/web.xml
```

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

### 4.12 Copy JasperServer WAR File to Tomcat

Now that the JasperServer source code has been built, you can manually deploy your jasperserver WAR file to your application server.

You can do the copy steps listed below.

Copy:

```
<js-src>/jasperserver/jasperserver-war/target/jasperserver - (copy whole directory)
```

To:

```
<tomcat>/webapps
```

### 4.13 Start JasperServer and Login

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

Login URL:

`http://localhost:8080/jasperserver`

Login credentials:

Username: jasperadmin - (Administrative user)

Password: jasperadmin



## 5 ADDITIONAL BUILDOMATIC INFORMATION

---

The buildomatic scripts automatically create a number of the key configurations needed by JasperServer. In this section, there is information on where to find some of the important buildomatic files.

### 5.1 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 JasperServer.

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

More generated property files:

```
<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:

```
js-ant clean-config
```

You can then regenerate them by running the target:

```
js-ant gen-config.
```

## 5.2 Existing and Generated Database Files

Buildomatic comes with files that support a number of databases. Some of the files are already complete. Other files are generated by, for instance, the `build-js-ddl-ce` target. Additionally, when the `build-js-ddl-ce` target is run, it overwrites the SQL file `../<db-type>/js-create.ddl`.

Here is where these files are found:

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

Here is an example of some of the key files:

```
mysql/js-create.ddl
```

```
mysql/quartz.ddl
```

```
mysql/sugarcrm.zip
```

```
mysql/supermart-update.sql
```

```
postgresql/js-create.ddl
```

```
postgresql/quartz.ddl
```

```
postgresql/sugarcrm.zip
```

```
postgresql/supermart-update.sql
```

## 5.3 Generated WAR File Location and deploy-webapp-ce Target

The JasperServer 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. The all occurs when the `build-ce` target is run. This location is the following:

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

When you later run the buildomatic target `deploy-webapp-ce` here are some of the actions which take place (using Tomcat as an example):

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

```
To: <tomcat>/webapps
```

```
Copy: <js-src>/jasperserver/buildomatic/build_conf/default/webapp/META-INF/context.xml
```

```
To: <tomcat>/webapps/jasperserver/META-INF
```

```
Copy: <js-src>/jasperserver/buildomatic/build_conf/default/webapp/WEB-INF/hibernate.properties
```

```
To: <tomcat>/webapps/jasperserver/WEB-INF/hibernate.properties
```

```
Copy: <js-src>/jasperserver/buildomatic/build_conf/default/webapp/WEB-INF/js.quartz.properties
```

```
To: <tomcat>/webapps/jasperserver/WEB-INF/js.quartz.properties
```

## APPENDIX A JAVA JVM SETTINGS

For more information on Java setting refer to the *JasperServer Installation Guide*.

### A.1 Java 1.5

The following tables layout recommended settings for Java 1.5.

#### A.1.1 General Java 1.5 JVM Settings

The following table recommends settings for Java 1.5. These settings can be increased or decreased for your particular requirements.

General Recommended Java JVM Settings:	
Tomcat file	<tomcat>/bin/setclasspath.bat or setclasspath.sh (or setenv.bat, setenv.sh)
JBoss file	<jboss>/bin/run.bat or run.sh
JAVA_OPTS setting	<pre>set JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx512m -XX:PermSize=32m      (Windows) -XX:MaxPermSize=128m -Xss2m set JAVA_OPTS=%JAVA_OPTS% -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:+CMSPermGenSweepingEnabled  JAVA_OPTS="\$JAVA_OPTS -Xms128m -Xmx512m -XX:PermSize=32m      (Linux) -XX:MaxPermSize=128m -Xss2m " JAVA_OPTS="\$JAVA_OPTS -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled -XX:+CMSPermGenSweepingEnabled "</pre>



You can cut and paste these settings from files in the <js-src>/jasperserver/scripts/java-settings directory.

### A.2 Java 1.6

If you are using Java 1.6 there are some additional JVM settings that enable all JasperServer features. Java 1.6 includes a web

services implementation that can conflict with JasperServer's AXIS-based web services classes. These conflicts could cause JasperServer web services and the resources that rely on them to fail (such as the JasperServer plugin for iReport, web services, and analysis XML/A connections).



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 table recommends settings for Java 1.6. These settings can be increased or decreased for your particular requirements.

General Recommended Java 1.6 JVM Settings:	
Tomcat file	<tomcat>/bin/setclasspath.bat or setclasspath.sh (or setenv.bat, setenv.sh)
JBoss file	<jboss>/bin/run.bat or run.sh
JAVA_OPTS setting	<div> <pre>set JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx512m -XX:PermSize=32m -XX:MaxPermSize=128m</pre> <p>(Windows)</p> <pre>set JAVA_OPTS=%JAVA_OPTS% -Xss2m -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled</pre> </div> <div> <pre>JAVA_OPTS="\$JAVA_OPTS -Xms128m -Xmx512m -XX:PermSize=32m -XX:MaxPermSize=128m "</pre> <p>(Linux)</p> <pre>JAVA_OPTS="\$JAVA_OPTS -Xss2m -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabled "</pre> </div>



You can cut and paste these settings from files in the <js-src>/jasperserver/scripts/java-settings directory.

### A.2.2 Additional Java 1.6 Web Services Settings

Java 1.6 includes a web services implementation that can conflict with JasperServer's AXIS based web services classes. In order to prevent these conflicts, add these special JVM settings.

Java JVM Settings for Java 1.6 Only:	
Tomcat file	<tomcat>/bin/setclasspath.bat or setclasspath.sh (or setenv.bat, setenv.sh)
JBoss file	<jboss>/bin/run.bat or run.sh

Java JVM Settings for Java 1.6 Only:		
JAVA_OPTS setting	<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>	(Windows)
	<pre> JAVA_OPTS="\$JAVA_OPTS -Djavax.xml.soap.MessageFactory=org.apache.axis.soap.Message FactoryImpl " 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.TransformerFac toryImpl " </pre>	(Linux)



You can cut and paste these settings from files in the <js-src>/jasperserver/scripts/java-settings directory.

If you do not utilize JasperServer web services or resources that use them, you do not need the settings in the table above.



## 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 JasperServer 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 required ant version 1.7.0 or higher. Additionally, the following jar needs to be included in the ant/lib directory:

ant-contrib.jar

If you are running with your own ant version then you can copy this jar from the following location to your ant/lib directory:

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

### B.2 Database Troubleshooting

The most common error encountered when building JasperServer involves the database. These errors often result from not being able to connect to the database. The Troubleshooting Appendix of the *JasperServer 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 2.0.9. The JasperServer source code build has been tested with Maven 2.0.9 and 2.1.0.

It is not recommended that versions older than 2.0.9 be used.



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

### B.3.2 Clear JasperServer 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 JasperServer 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 JasperServer artifacts:

1. Change directories:

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

2. Remove the old versions:

```
remove com/jaspersoft      (OS 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 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/
jasperintel/jasperserver
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 JAR File

If you get an error indicating the Maven wasn't able to transfer a particular file, this is because Maven has to download many different resources on the first build. In the following example, there was a transfer error on the `castor.jar` file. In this case, you may receive an error similar to:

```
[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:\workspaces\src\jasperserver-repo\jasperserver-maven)
```

Such problems can be fixed by re-running the `mvn install` command (that is, restart the build). To avoid these errors altogether, use the prepopulated Maven repository when you 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.acegisecurity:acegi-security:jar:1.0.0
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:\workspace\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.

## B.4 Other Build Troubleshooting

### B.4.1 Error When Building Database Scripts

If you get an error when compiling in the following directory:

`jasperserver-repository-hibernate/build-db`

It may be because the build step did not find the Quartz scripts. The Quartz scripts are located in the `<js-src>` directory in the following location:

`<js-src>/jasperserver/scripts/quartz`

The error message in this case is typically:

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

If you get this error, the most likely problem is that your `.m2/settings.xml` doesn't correctly point to your source location:

`<js.quartz.script>/home/user/js-src/jasperserver/scripts/quartz/tables_mysql_innodb.sql</js.quartz.script>`

Ensure that the line above evaluates correctly to the `scripts/quartz` directory.

## 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.



## APPENDIX C BUILDING FROM JASPERFORGE.ORG PUBLIC SVN SOURCE

---

The JasperServer source code hosted on the JasperForge.org site is publicly available and is managed by Subversion source control (SVN). This section describes the checkout and build of the public source.

### Browse Source Code on JasperForge.org

You can go to the JasperServer Source Code page on the JasperForge.org site and browse the source code. The following link will go directly to the Source Code page:

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

Click the `jasperserver` link to browse the source files.

### List the Source Code

You can get a listing of the JasperServer source code with the following command:

```
svn list --username anonsvn http://jasperforge.org/svn/repos/jasperserver
password: anonsvn
```

### Checkout the Source Code

Under the `jasperserver` source directory are the three standard Subversion subdirectories: `branches`, `tags`, and `trunk`. If you check out at the top level you will get all source code including all branches and tags. It is better to narrow the checkout command.

To checkout the most current trunk HEAD code:

```
svn checkout --username anonsvn http://jasperforge.org/svn/repos/jasperserver/trunk jasperserver-trunk
password: anonsvn
```

To checkout a specific tagged version (such as 3.5.0 GA):

```
svn checkout --username anonsvn http://jasperforge.org/svn/repos/jasperserver/tags/js-3.5.0 jasperserver-3.5.0
password: anonsvn
```

### Build the Source Code

The Maven build tool will need to be configured in order to compile the source code. See section [4.1, “Manual Setup of JasperServer Build,” on page 17](#) for information on setting up Maven in order to run with the JasperServer source code.

You can get your maven `settings.xml` from the following location:

```
<js-src>/scripts/dev-setup/settings.xml
```

Make sure that your Maven settings.xml file has the following definitions:

```
<repository>
  <id>JasperForge Maven Repository</id>
  <!-- note matching <server> def at bottom of file for anonymous access -->
  <url>http://jasperforge.org/svn/repos/maven2</url>
  <snapshots>
    <enabled>true</enabled>
  </snapshots>
  <releases>
    <enabled>true</enabled>
  </releases>
</repository>
```

```
<server>
  <id>JasperForge Maven Repository</id>
  <!-- note: matches <repository> def above -->
  <username>anonsvn</username>
  <password>anonsvn</password>
</server>
```

### <http://jasperforge.org/svn/repos/maven2>

The JasperForge.org site maintains a remote Maven repository that holds dependencies that might be hard to find out in the standard public maven repositories.

The JasperServer public source code should should always be able to be built and any new JasperServer dependencies should be added to the JasperForge Maven repository. If you have trouble resolving a 3rd party dependency you should report the problem on the JasperForge.org Forum pages.

### Build JasperServer Source Code

Once your Maven is fully configured, based on the setup steps in section [4.1, “Manual Setup of JasperServer Build,” on page 17](#), you can compile the source code. Here are the basic steps:

```
cd <js-src>/jasperserver
mvn clean install                                     creates the war file
cd <js-src>/jasperserver/jasperserver-repository-hibernate/build_db
mvn clean install
cd <js-src>/jasperserver/jasperserver-unit-test
mvn clean install
```

The first `mvn clean install` command builds the jasperserver WAR file. If everything compiled cleanly, you can find this WAR file in `<js-src>/jasperserver/jasperserver-war/target`.

## GLOSSARY

---

### **Ad Hoc Editor**

JasperServer's integrated report designer. Starting from a collection of fields predefined in a Topic or selected from a Domain, the Ad Hoc Editor lets you drag and drop report elements to draft, preview, and finalize reports. Like JRXML reports, Ad Hoc reports can be run, printed, and scheduled within JasperServer. In addition, Ad Hoc reports may be reopened in the Ad Hoc Editor, further modified, and saved.

### **Analysis Client Connection**

A definition for retrieving an analysis view. An analysis client connection is either a direct Java connection (Mondrian connection) or an XML-based API connection (XMLA connection).

### **Analysis Schema**

A metadata definition of a multidimensional database. In JasperAnalysis, schemas are stored in the repository as XML file resources.

### **Analysis View**

A view of multidimensional data that is based on an analysis client connection and an MDX query. It is the entry point to analysis operations, such as slice and dice, drill down, and drill through.

### **Calculated Field**

In a Domain, a field whose value is calculated from a user-written formula that may include any number of fields, operators, and constants. A calculated field is defined in the Domain Designer dialog, and it becomes one of the items to which the Domain's security file and locale bundles can apply.

### **CRM**

Customer Relationship Management. The practice of managing every facet of a company's interactions with its clientele. CRM applications help businesses track and support their customers.

### **CrossJoin**

An MDX function that combines two or more dimensions into a single axis (column or row).

### **Cube**

The basis of most analysis applications, a cube is a data structure that contains three or more dimensions that categorize the cube's quantitative data. When you navigate the data displayed in an analysis view, you are exploring a cube.

## **Custom Field**

In the Ad Hoc Editor, a field that is created through menu items as a simple function of one or two available fields, including other custom fields. When a custom field becomes too complex or needs to be used in many reports, it is best to define it as a calculated field in a Domain.

## **Dashboard**

A collection of reports, input controls, graphics, labels, and web content displayed in a single, integrated view. Dashboards often present a high level view of your data, but input controls can parameterize the data to display. For example, you can narrow down the data to a specific date range. Embedded web content, such as other web-based applications or maps, make dashboards more interactive and functional.

## **Derived Table**

In a Domain, a derived table is defined by an additional query whose result becomes another set of items available in the Domain. For example, with a JDBC data source, you can write an SQL query that includes complex functions for selecting data. You can use the items in a derived table for other operations on the Domain, such as joining tables, defining a calculated field, or filtering. The items in a derived table can also be referenced in the Domain's security file and locale bundles.

## **Data Policy**

In JasperServer, a setting that determines how JasperServer should process and cache data used by Ad Hoc reports. Select your data policies by clicking **Manage > Ad Hoc Options**.

## **Data Source**

Defines the connection properties that JasperServer needs to access data. JasperServer transmits queries to data sources and obtains datasets in return for use in filling reports and previewing Ad Hoc reports. JasperServer supports JDBC, JNDI, and Bean data sources; custom data sources can be defined as well.

## **Dataset**

A collection of data arranged in columns and rows. Datasets are equivalent to relational results sets and the `JRDataSource` type in JasperReports.

## **Datatype**

In JasperServer, a datatype is used to characterize a value entered through an input control. A datatype must be of type text, number, date, or date-time. It can include constraints on the value of the input, for example maximum and minimum values. As such, a JasperServer datatype is more structured than a datatype in most programming languages.

## **Denormalize**

A process for creating table joins that speeds up data retrieval at the cost of having duplicate row values between some columns.

## **Dice**

An OLAP operation to select columns.

## **Dimension**

A categorization of the data in a cube. For example, a cube that stores data about sales figures might include dimensions such as time, product, region, and customer's industry.

## **Domain**

A virtual view of a data source that presents the data in business terms, allows for localization, and provides data-level security. A Domain is not a view of the database in relational terms, but it implements the same functionality within JasperServer. The design of a Domain specifies tables in the database, join clauses, calculated fields, display names, and default properties, all of which define items and sets of items for creating Ad Hoc reports.

## **Domain Topic**

A Topic that is created from a Domain by the Choose Ad Hoc Data wizard. A Domain Topic is based on the data source and items in a Domain, but it allows further filtering, user input, and selection of items. Unlike a JRXML-based Topic, a Domain Topic can be edited in JasperServer by users with the appropriate permissions.



**Drill**

To click on an element of an analysis view to change the data that is displayed:

- Drill down. An OLAP operation that exposes more detailed information down the hierarchy levels by delving deeper into the hierarchy and updating the contents of the navigation table.
- Drill through. An OLAP operation that displays detailed transactional data for a given aggregate measure. Click a fact to open a new table beneath the main navigation table; the new table displays the low-level data that constitutes the data that was clicked.
- Drill up. An OLAP operation for returning the parent hierarchy level to view to summary information.

**Eclipse**

An open source Integrated Development Environment (IDE) for Java and other programming languages, such as C/C++.

**ETL**

Extract, Transform, Load. A process that retrieves data from transactional systems, and filters and aggregates the data to create a multidimensional database.

**Fact**

The specific value or aggregate value of a measure for a particular member of a dimension. Facts are typically numeric.

**Field**

A field is equivalent to a column in the relational database model. Fields originate in the structure of the data source, but you may define calculated fields in a Domain or custom fields in the Ad Hoc Editor. Any type of field, along with its display name and default formatting properties, is called an item and may be used in the Ad Hoc editor.

**Frame**

A dashboard element that displays reports or custom URLs. Frames can be mapped to input controls if their content can accept parameters.

**Group**

In a report, a group is a set of data rows that have an identical value in a designated field.

- In a table, the value appears in a header and footer around the rows of the group, while the other fields appear as columns.
- In a chart, the field chosen to define the group becomes the independent variable on the X axis, while the other fields of each group are used to compute the dependent value on the Y axis.

**Hierarchy Level**

In analysis, a member of a dimension containing a group of members.

**Input Control**

A button, check box, drop-down list, text field, or calendar icon that allows users to enter a value when running a report or viewing a dashboard that accepts input parameters. For JRXML reports, input controls and their associated datatypes must be defined as repository objects and explicitly associated with the report. For Domain-based reports that prompt for filter values, the input controls are defined internally. When either type of report is used in a dashboard, its input controls are available to be added as special content.

**Item**

When designing a Domain or creating a Topic based on a Domain, an item is the representation of a database field or a calculated field along with its display name and formatting properties defined in the Domain. Items can be grouped in sets and are available for use in the creation of Ad Hoc reports.

**JavaBean**

A reusable Java component that can be dropped into an application container to provide standard functionality.

**JDBC**

Java Database Connectivity. A standard interface that Java applications use to access databases.

## **JNDI**

Java Naming and Directory Interface. A standard interface that Java applications use to access naming and directory services.

## **Join Tree**

In Domains, a collection of joined tables from the actual data source. A join is the relational operation that associates the rows of one table with the rows of another table based on a common value in given field of each table. Only the fields in a same join tree or calculated from the fields in a same join tree may appear together in a report.

## **JPivot**

An open source graphical user interface for OLAP operations. For more information, visit <http://jpivot.sourceforge.net/>.

## **MDX**

Multidimensional Expression Language. A language for querying multidimensional objects, such as OLAP (On Line Analytical Processing) cubes, and returning cube data for analytical processing. An MDX query is the query that determines the data displayed in an analysis view.

## **Measure**

Depending on the context:

- In a report, a formula that calculates the values displayed in a table's columns, a crosstab's data values, or a chart's dependent variable (such as the slices in a pie).
- In an analysis view, a formula that calculates the facts that constitute the quantitative data in a cube.

## **Mondrian**

A Java-based, open source multidimensional database application.

## **Mondrian Connection**

An analysis client connection that consists of an analysis schema and a data source used to populate an analysis view.

## **Mondrian Schema Editor**

An open source Eclipse plugin for creating Mondrian analysis schemas.

## **Mondrian XMLA Source**

A server-side XMLA source definition of a remote client-side XMLA connection used to populate an analysis view using the XMLA standard.

## **MySQL**

An open source relational database management system. For information, visit <http://www.mysql.com/>.

## **Navigation Table**

The main table in an analysis view that displays measures and dimensions as columns and rows.

## **Object**

In JasperServer, anything residing in the repository, such as an image, file, font, data source, topic, domain, report element, saved report, report output, dashboard, or analysis view. The folders that contain repository objects are also objects. Administrators set user and role-based access privileges on repository objects to establish a security policy.

## **OLAP**

On Line Analytical Processing. Provides multidimensional views of data that help users analyze current and past performance and model future scenarios.

## **Organization**

A set of users that share resources and repository objects in JasperServer. An organization has its own user accounts, roles, and root folder in the repository to securely isolate it from other organizations that may be hosted on the same instance of JasperServer.

**Organization Admin**

Also called the organization administrator. A user in an organization with the privileges to manage the organization's user accounts and roles, repository permissions, and repository content. An organization admin can also create sub-organizations and manage all of their accounts, roles, and repository objects. The default organization admin in each organization is the `jasperadmin` account.


**Outlier**

A fact that seems incongruous when compared to other member's facts. For example, a very low sales figure or a very high number of helpdesk tickets. Such outliers may indicate a problem (or an important achievement) in your business. JasperAnalysis excels at revealing outliers.

**Parameter**

Named values that are passed to the engine at report-filling time to control the data returned or the appearance and formatting of the report. A report parameter is defined by its name and type. In JasperServer, parameters can be mapped to input controls that users can interact with.

**Pivot**

To rotate a crosstab such that its row groups become column groups and its column groups become rows. In the Ad Hoc Editor, pivot the crosstab by clicking .

**Pivot Table**

A table with two physical dimensions (for example, X and Y axis) for organizing information containing more than two logical dimensions (for example, PRODUCT, CUSTOMER, TIME, and LOCATION), such that each physical dimension is capable of representing one or more logical dimensions, where the values described by the dimensions are aggregated using a function such as SUM.

Pivot tables are used in JasperAnalysis.

**Properties**

Settings associated with an object. The settings determine certain features of the object, such as its color and label. Properties are normally editable. In Java, properties can be set in files listing objects and their settings.

**Repository**

The tree structure of folders that contain all saved reports, dashboards, analysis views, and resources. Users access the repository through the JasperServer web interface or through iReport. Applications can access the repository through the web service API. Administrators use the import and export utilities to back up the repository contents.

**Role**

A security feature of JasperServer. Administrators create named roles, assign them to user accounts, and then set access permissions to repository objects based on those roles. JasperServer also makes certain functionality available to users based on their roles, which determines certain menu options displayed to those users.

**Schema**

A logical model that determines how data is stored. For example, the schema in a relational database is a description of the relationships between tables, views, and indexes. In JasperAnalysis, an OLAP schema is the logical model of the data that appears in an analysis view; they are uploaded to the repository as resources. For Domains, schemas are represented in XML design files.

**Set**

In Domains and Domain Topics, a named collection of items grouped together for ease of use in the Ad Hoc Editor. A set can be based on the fields in a table or entirely defined by the Domain creator, but all items in a set must originate in the same join tree. The order of items in a set is preserved.

**Slice**

An OLAP operation for filtering data rows.

## SQL

Structured Query Language. A standard language used to access and manipulate data and schemas in a relational database.

## System Admin

Also called the system administrator. A user who has unlimited access to manage all organizations, users, roles, repository permissions, and repository objects across the entire JasperServer instance. The system admin can create root-level organizations and manage all server settings. The default system admin is the `superuser` account.

## Topic

A JRXML file created externally and uploaded to JasperServer as a basis for Ad Hoc reports. Topics are created by business analysts to specify a data source and a list of fields with which business users can create reports in the Ad Hoc Editor. Topics are stored in the Ad Hoc Components folder of the repository and displayed when a user launches the Ad Hoc Editor.

## Transactional Data

Data that describe measurable aspects of an event, such as a retail transaction, relevant to your business. Transactional data are often stored in relational databases, with one row for each event and a table column or field for each measure.

## User

Depending on the context:

- A person who interacts with JasperServer to fulfill a goal. There are generally three categories of users: administrators who install and configure JasperServer, database experts or business analysts who create data sources and Domains, and business users who create and view reports and dashboards.
- A user account created for a specific person or purpose. The account associates the login name with user's full name, password, and email address. Roles are assigned to user accounts to determine access to objects in the repository.

## WCF

Web Component Framework. A low-level GUI component of JPivot. For more information, see <http://jpivot.sourceforge.net/wcf/index.html>.

## XML

eXtensible Markup language. A standard for defining, transferring, and interpreting data for use across any number of XML-enabled applications.

## XML/A

XML for Analysis. An XML standard that uses Simple Object Access protocol (SOAP) to access remote data sources. For more information, see <http://www.xmla.org/>

## XML/A Connection

A type of analysis client connection that consists of Simple Object Access Protocol (SOAP) definitions used to populate an analysis view.