JasperReports® Server Community Project Upgrade Guide

Release 6.4

http://www.jaspersoft.com

This is version 0217-JSO64-08 of the *JasperReports Server Community Project Upgrade Guide*.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

TIBCO JasperReports® Server builds on TIBCO JasperReports® Library 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 provide shared services, such as security, a repository, and scheduling. The server exposes comprehensive public interfaces enabling seamless integration with other applications and the capability to easily add custom functionality.

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

The heart of the TIBCO Jaspersoft® BI Suite is the server, which provides the ability to:

- Easily create new reports based on views designed in an intuitive, web-based, drag and drop Ad Hoc Editor.
- Efficiently and securely manage many reports.
- Interact with reports, including sorting, changing formatting, entering parameters, and drilling on data.
- Schedule reports for distribution through email and storage in the repository.
- Arrange reports and web content to create appealing, data-rich Jaspersoft Dashboards that quickly convey business trends.

For users interested in multi-dimensional modeling, we offer Jaspersoft® OLAP, which runs as part of the server.

While the Ad Hoc Editor lets users create simple reports, more complex reports can be created outside of the server. You can either use Jaspersoft® Studio or manually write JRXML code to create a report that can be run in the server. We recommend that you use Jaspersoft Studio unless you have a thorough understanding of the JasperReports file structure.

You can use the following sources of information to learn about JasperReports Server:

- Our core documentation describes how to install, administer, and use JasperReports Server and Jaspersoft Studio. Core documentation is available as PDFs in the doc subdirectory of your JasperReports Server installation. You can also access PDF and HTML versions of these guides online from the Documentation section of the Jaspersoft Community website.
- Our Ultimate Guides document advanced features and configuration. They also include best practice recommendations and numerous examples. You can access PDF and HTML versions of these guides online from the Documentation section of the Jaspersoft Community website.

- Our Online Learning Portal lets you learn at your own pace, and covers topics for developers, system administrators, business users, and data integration users. The Portal is available online from the Professional Services section of our website.
- Our free samples, which are installed with JasperReports Library, Jaspersoft Studio, and JasperReports Server, are available and documented online. Please visit our GitHub repository.
- If you have a subscription to our professional support offerings, please contact our Technical Support team when you have questions or run into difficulties. They're available on the web at and through email at http://support.tibco.com and js-support@tibco.com.

JasperReports Server is a component of both a community project and commercial offerings. Each integrates the standard features such as security, scheduling, a web services interface, and much more for running and sharing reports. Commercial editions provide additional features, including Ad Hoc views and reports, advanced charts, dashboards, Domains, auditing, and a multi-organization architecture for hosting large BI deployments.

1.1 Server Upgrade Distributions

The following distribution package is available for JasperReports Server upgrade:

Distribution Package	Description
WAR File Distribution Zip	Supports upgrade from version 4.5 or later. Supports all certified application servers. Supports all certified repository databases. Supports Windows, Linux, Mac, and other platforms. File name is: **TIB_js-jrs_cp_6.4.0_bin.zip**

1.1.1 About Bundled Apache Ant

We recommend Apache Ant version 1.9.4, which is bundled with the Overlay Upgrade ZIP and the War File Distribution ZIP. The Ant scripts used for upgrade come with Windows and Linux batch scripts pre-configured to use the bundled version of Apache Ant.

If you want to run your own version of Apache Ant, version 1.8.1 or later is required.

The bundled Apache Ant includes an additional jar This jar (ant-contrib.jar) enables conditional logic in Ant. If you're running your own Ant you should copy the ant-contrib.jar to your <Ant_HOME>/lib folder.

On Linux and Solaris, the Ant commands may not be compatible with all shells. If you get errors, use the `bash` shell explicitly. For more information, see the information on the bash shell in the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.

CHAPTER 2 UPGRADING FROM 6.3 TO 6.4

This chapter describes the recommended procedure for upgrading to JasperReports Server 6.4 from version 6.3. The examples show you how to upgrade using the js-upgrade shell scripts.

This chapter contains the following sections:

2.1 Upgrade Steps Overview

These are the general steps used in this section:

1. Identify your customizations.
2. Back up your current JasperReports Server instance.
3. Download and set up the new 6.4 JasperReports Server WAR file distribution zip.
4. Run the js-upgrade script as described in **2.6, "Upgrading to JasperReports Server 6.4," on page 10**.

If your current instance of JasperReports Server has modifications or extensions, keep track of these and re-integrate them into your 6.4 instance after upgrading.

2.2 Upgrading with Customizations

If your current instance of JasperReports Server has modifications or extensions, keep track of these and re-integrate them into your 6.4 instance after upgrading. See **Appendix A, "Planning Your Upgrade ," on page 25** to determine if any customizations you've made to your existing version of JasperReports Server are affected by changes to the updated version.

2.3 Back Up Your JasperReports Server Instance

First back up your JasperReports Server WAR file and `jasperserver` database so you can restore them if necessary. Perform these steps from the command line in a Windows or Linux shell.

This backup example is for Tomcat with the PostgreSQL or MySQL database. For other databases, consult your DB administration documentation for backup information.

Back up your JasperReports Server War File:

1. Create a folder where you can save your `jasperserver` war file. For example, `C:\JS_BACKUP` or `/opt/JS_BACKUP`.
2. Copy `<tomcat>/webapps/jasperserver` to `<path>/JS_BACKUP`

Back up your jasperserver Database:

1. Create a folder (if you did not do so in the step above) where you can save your `jasperserver` database, For example, `C:\JS_BACKUP` or `/opt/JS_BACKUP`.
2. Run the following commands for PostgreSQL or MySQL:
   - PostgreSQL

     ```
     cd <path>/JS_BACKUP
     pg_dump --username=postgres  jasperserver  >  js-db-dump.sql
     ```

   - MySQL

     ```
     cd <path>/JS_BACKUP
     ```

     Windows:
     ```
     mysqldump --user=root --password=<password> jasperserver > js-db-dump.sql
     ```
     Linux:
     ```
     mysqldump --user=root --password=<password> --host=127.0.0.1 jasperserver >
     js-db-dump.sql
     ```

For MySQL, If you receive an error about packet size, see the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.

2.4 Preparing the JasperReports Server 6.4 WAR File Distribution

Use the buildomatic `js-upgrade` scripts included in the 6.4 WAR file distribution ZIP release package to carry out the upgrade. The WAR file distribution comes in a compressed ZIP file named `TIB_js-jrs_cp_6.4.0_bin.zip`.

Follow these steps to obtain and unpack the WAR file distribution ZIP file:

1. Download the WAR file distribution from the Jaspersoft community site (http://community.jaspersoft.com).
2. Extract all files from `TIB_js-jrs_cp_6.4.0_bin.zip`. Choose a destination, such as a `C:\Jaspersoft` folder on Windows, `/home/<user>` on Linux, or `/Users/<user>` on Mac.

After you unpack the WAR File Distribution, the resulting location will be known as:

`<js-install-6.4>`

2.5 Configuring Buildomatic for Your Database and Application Server

This upgrade procedure uses the `js-upgrade-samedb-ce` shell script.

For Unix, the bash shell is required for the js-upgrade scripts. If you're installing to a non-Linux Unix platform such as HP-UX, IBM AIX, FreeBSD or Solaris, you need to download and install the bash shell. See the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide* for more information.

This section shows example configurations for the PostgreSQL and MySQL databases.

2.5.1 Example Buildomatic Configuration

The `default_master.properties` file handles the upgrade configuration. We provide a sample configuration file for each database. You must specify your database credentials and application server location, and rename the file to `default_master.properties`.

2.5.1.1 PostgreSQL Example

To configure `default_master.properties` for PostgreSQL:

1. Locate the `postgresql_master.properties` sample configuration file:

Database	Master Properties File
PostgreSQL	<js-install-6.4>/buildomatic/sample_conf/postgresql_master.properties

2. Copy the file to `<js-install-6.4>/buildomatic`
3. Rename the file `default_master.properties`
4. Edit `default_master.properties` for your database and application server:

Database	Sample Property Values

PostgreSQL	`appServerType=tomcat6 [tomcat7, jboss, glassfish2, glassfish3]`
	`appServerDir=c:\\Apache Software Foundation\\Tomcat 7`
	`dbUsername=postgres`
	`dbPassword=postgres`
	`dbHost=localhost`

2.5.1.2 MySQL Example

To configure `default_master.properties` for MySQL:

1. Locate the `mysql_master.properties` sample configuration file:

Database	Master Properties File
MySQL	<js-install-6.4>/buildomatic/sample_conf/mysql_master.properties

2. Copy the file to `<js-install-6.4>/buildomatic`
3. Rename the file `default_master.properties`
4. Edit `default_master.properties` for your database and application server.

Database	Sample Property Values
MySQL	appServerType=tomcat6 [tomcat7, jboss, glassfish2, glassfish3] appServerDir=C:\\Apache Software Foundation\\Tomcat 7 dbUsername=root dbPassword=password dbHost=localhost

2.6 Upgrading to JasperReports Server 6.4

Now that your buildomatic scripts are configured, you can complete the upgrade.

Make sure you've backed up your `jasperserver` database before proceeding.

Make sure you've backed up your old JasperReports Server WAR file before proceeding.

1. Stop your application server
2. Start your database server
3. Run the following commands:

Commands	Description
`cd <js-install-6.4>/buildomatic`	
`js-upgrade-samedb-ce.bat`	(Windows) Upgrade jasperserver war file, upgrade jasperserver database to 6.4, add 6.4 repository resources into the database
`./js-upgrade-samedb-ce.sh`	(Linux) Upgrade jasperserver war file, upgrade jasperserver database to 6.4, add 6.4 repository resources into the database

2.6.1 js-upgrade Test Mode

Use the `test` option to run the `js-upgrade` script in test mode. For example, on Windows, enter:

```
cd <js-install-6.4>/buildomatic
js-upgrade-samedb-ce.bat test
```

In test mode, the js-upgrade scripts check your default_master.properties settings and validate your application server location and its ability to connect to your database. Test mode can help you debug issues like an incorrect database password without altering your system.

2.6.2 Output Log Location

The js-upgrade script creates an output log that captures both standard and error output. If problems occur during script execution, or you just want to remember which options you chose, open the output log file located

here:

```
<js-install-6.4>/buildomatic/logs/js-upgrade-<date>-<number>.log
```

2.6.3 Errors

If you encounter errors running the `js-upgrade` script, first look at the output log to see if you can spot the errors. For help, refer to the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*. The information in this appendix applies to both `js-upgrade` scripts and `js-install` scripts.

If you need to modify values in your `default_master.properties` file, you can simply edit the file. When you run the `js-upgrade` script again, it uses the new values.

2.7 Starting and Logging into JasperReports Server6.4

Start your application server. Your database should already be running.

2.7.1 Clearing Your Browser Cache

Before you log in, make sure you and your end users clear the browser cache. JavaScript files, which enable the UI elements of JasperReports Server, are typically cached by the browser. Clear the cache to ensure that the newer files are used.

2.7.2 Logging into JasperReports Server

Log in using the following URL, user IDs, and passwords:

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

User ID	Password	Description
`jasperadmin`	`<your-password>`	Administrator for the default organization

Your JasperReports Server instance has now been upgraded to 6.4. If you have startup or login problems, refer to the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.

2.8 Additional Tasks to Complete the Upgrade

Perform these tasks with the application server shutdown.

2.8.1 Handling JasperReports Server Customizations

If you made modifications to the original JasperReports Server application, these configurations are typically found in the `WEB-INF/applicationContext-*.xml` set of files.

You'll need to manually copy configuration changes, like client-specific security classes or LDAP server configurations, from your previous environment and integrate them with your upgraded environment.

2.8.2 Clearing the Application Server Work Folder

Application servers have work folders where JasperReports Server files are compiled and cached and other objects are stored. When you update the WAR file, the buildomatic `deploy-webapp-ce` target should automatically clear the application server's `work` directory, but it's a good practice to double-check. A permission problem, or some other problem, could prevent the clearing of the work folder.

To clear the work folder in Tomcat:

1. Change directory to `<tomcat>/work`.
2. Delete all the files and folders in this directory.

2.8.3 Clearing the Application Server Temp Folder

JasperReports Server uses caching to speed operations within the application. Caching files are created and stored in the application server, usually in a `temp` folder. Clear this `temp` folder to avoid any post-upgrade conflicts. Typically, the `temp` folder used by an application server corresponds to the path referenced by the `java.io.tmpdir` Java system property. For Apache Tomcat the `temp` folder is `<tomcat>/temp`.

To clear the temp folder in Apache Tomcat:

1. Change directory to `<tomcat>/temp`
2. Delete all the files and folders in this directory

2.8.4 Clearing the Repository Cache Database Table

In the `jasperserver` database, compiled JasperReports Library resources are cached in the `JIRepositoryCache` table for increased efficiency at runtime. Because the JasperReports Library JAR is typically updated with each new release, old cached items can get out of date and cause errors at runtime. If you encounter errors that mention a JasperReports Library "local class incompatible," check your repository cache table. In summary, you can clear your `jasperserver` database cache table as part of this upgrade process whether or not there are errors.

To manually clear the repository cache database table, run a SQL command similar to one shown below:

```
update JIRepositoryCache set item_reference = null;
delete from JIRepositoryCache;
```

2.9 Old Manual Upgrade Steps: 6.3 to 6.4

This section describes the older, manual upgrade steps used before we implemented the `js-upgrade` shell scripts in release 4.0. They're provided here mainly as a reference for internal use.

We recommend using the `js-upgrade` shell scripts described in the beginning of this chapter instead of these manual commands:

Commands	Description
`cd <js-install-6.4>/buildomatic`	

Commands	Description
`js-ant upgrade-6.3-6.4-ce`	Execute SQL script to upgrade database to 6.4. Execute script buildomatic/install_ resources/sql/<dbType>/upgrade-<dbType>-6.3.0-6.4.0-ce.sql
`js-ant import-minimal-for-upgrade-ce`	Load themes and other core resources for 6.4. Note: "import-minimal-for-upgrade" will import core resources in an "update" mode so that the older 6.3 core resources will be overwritten. Additionally, the "skip-user-update" option will be applied so that jasperadmin users will not have their passwords modified.
`js-ant import-sample-data-upgrade-ce`	(Optional) Load the 6.4 sample data.
`js-ant deploy-webapp-ce`	Delete old 6.3 war file, deploy the 6.4 war file.

CHAPTER 3 UPGRADING FROM 4.5 - 6.2 TO 6.4

This chapter describes the recommended procedure for upgrading from JasperReports Server 4.5 through 6.2 to JasperReports Server 6.4. If you're upgrading from version 6.3 to 6.4, we recommend the procedure in **Chapter 2, "Upgrading from 6.3 to 6.4," on page 7**.

This upgrade procedure uses the JasperReports Server WAR File Distribution ZIP release package and the included buildomatic scripts. Our examples are for upgrading from version 6.2.

This chapter contains the following sections:

3.1 Upgrade Steps Overview

These are the general steps used in this section:

1. Plan your upgrade.
2. Back up your current JasperReports Server instance.
3. Export your existing repository data. For example, export your 6.2 data.
4. Download and set up the new 6.4 JasperReports Server WAR file distribution zip.
5. Run the js-upgrade script as described in **3.7, "Upgrading to JasperReports Server 6.4," on page 19**.

3.2 Upgrading with Customizations

If your current instance of JasperReports Server has modifications or extensions, keep track of these and re-integrate them into your 6.4 instance after upgrading. See **Appendix A, "Planning Your Upgrade ," on**

page 25 to determine if any customizations you've made to your existing version of JasperReports Server are affected by changes to the updated version.

3.3 Back Up Your JasperReports Server Instance

First back up your JasperReports Server WAR file and `jasperserver` database so you can restore them if necessary. Perform these steps from the command line in a Windows or Linux shell.

This backup example is for Tomcat with the PostgreSQL or MySQL database. For other databases, consult your DB administration documentation for backup information.

Back up your JasperReports Server War File:

1. Create a folder where you can save your `jasperserver` war file. For example, `C:\JS_BACKUP` or `/opt/JS_BACKUP`.
2. Copy `<tomcat>/webapps/jasperserver` to `<path>/JS_BACKUP`

Back up your jasperserver Database:

1. Create a folder (if you did not do so in the step above) where you can save your `jasperserver` database, For example, `C:\JS_BACKUP` or `/opt/JS_BACKUP`.
2. Run the following commands for PostgreSQL or MySQL:
   - PostgreSQL

     ```
     cd <path>/JS_BACKUP
     pg_dump --username=postgres jasperserver > js-db-dump.sql
     ```

   - MySQL

     ```
     cd <path>/JS_BACKUP
     ```

     Windows:
     ```
     mysqldump --user=root --password=<password> jasperserver > js-db-dump.sql
     ```

     Linux:
     ```
     mysqldump --user=root --password=<password> --host=127.0.0.1 jasperserver >
     js-db-dump.sql
     ```

For MySQL, If you receive an error about packet size, see the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.

3.4 Exporting Current Repository Data

You need to export your old repository data. Use the JasperReports Server export utility to export using:

- the buildomatic scripts (if you originally installed using buildomatic).
- the `js-export-ce.bat/.sh` script found in the `<js-install>/buildomatic` folder.

3.4.1 Using Buildomatic Scripts to Export Data

If you configured buildomatic and your `default_master.properties` file for export as described in the JasperReports Server Community Project Administrator Guide, you can use buildomatic to export your repository data. For example, to export 6.2 repository data, use the following commands:

1. Navigate to the buildomatic directory:

   ```
   cd <js-install-6.2>/buildomatic
   ```

2. Run buildomatic with the export target:

Windows: `js-ant.bat export-everything-ce -DexportFile=js-6.2-export.zip`

Linux: `./js-ant export-everything-ce -DexportFile=js-6.2-export.zip`

Note the location of this export file so that you can use it during the 6.4 upgrade process.

3.4.2 Using the js-export Script to Export Data

To use the `js-export-ce.bat/.sh` script, navigate to the buildomatic folder, for example, `<js-install-6.2>/buildomatic`. If you're using the PostgreSQL database the js-export script should already be configured to run. If you're using a different database, or you've changed database passwords, you may need to update the `js-export` configuration.

Run the following commands:

1. Navigate to the buildomatic directory:

```
cd <js-install-6.2>/buildomatic
```

2. Run the `js-export` script:

Windows: `js-export-ce.bat --everything --output-zip js-6.2-export.zip`

Linux: `js-export-ce.sh --everything --output-zip js-6.2-export.zip`

Note the location of this export file so that you can use it during the 6.4 upgrade process.

3.5 Preparing the JasperReports Server 6.4 WAR File Distribution

Use the buildomatic `js-upgrade` scripts included in the 6.4 WAR file distribution ZIP release package to carry out the upgrade. The WAR file distribution comes in a compressed ZIP file named `TIB_js-jrs_cp_6.4.0_bin.zip`.

Follow these steps to obtain and unpack the WAR file distribution ZIP file:

1. Download the WAR file distribution from the Jaspersoft community site (http://community.jaspersoft.com).
2. Extract all files from `TIB_js-jrs_cp_6.4.0_bin.zip`. Choose a destination, such as a `C:\Jaspersoft` folder on Windows, `/home/<user>` on Linux, or `/Users/<user>` on Mac.

After you unpack the WAR File Distribution, the resulting location will be known as:

```
<js-install-6.4>
```

3.6 Configuring Buildomatic for Your Database and Application Server

This upgrade procedure uses the `js-upgrade-newdb-ce` shell script.

For Unix, the bash shell is required for the js-upgrade scripts. If you're installing to a non-Linux Unix platform such as HP-UX, IBM AIX, FreeBSD or Solaris, you need to download and install the bash shell. See the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide* for more information.

This section shows example configurations for the PostgreSQL and MySQL databases.

3.6.1 Example Buildomatic Configuration

The `default_master.properties` file handles the upgrade configuration. We provide a sample configuration file for each database. You must specify your database credentials and application server location, and rename the file to `default_master.properties`.

3.6.1.1 PostgreSQL Example

To configure `default_master.properties` for PostgreSQL:

1. Locate the `postgresql_master.properties` sample configuration file:

Database	Master Properties File
PostgreSQL	<js-install-6.4>/buildomatic/sample_conf/postgresql_master.properties

2. Copy the file to `<js-install-6.4>/buildomatic`
3. Rename the file `default_master.properties`
4. Edit `default_master.properties` for your database and application server:

Database	Sample Property Values

PostgreSQL	`appServerType=tomcat6 [tomcat7, jboss, glassfish2, glassfish3]`
	`appServerDir=c:\\Apache Software Foundation\\Tomcat 7`
	`dbUsername=postgres`
	`dbPassword=postgres`
	`dbHost=localhost`

3.6.1.2 MySQL Example

To configure `default_master.properties` for MySQL:

1. Locate the `mysql_master.properties` sample configuration file:

Database	Master Properties File
MySQL	<js-install-6.4>/buildomatic/sample_conf/mysql_master.properties

2. Copy the file to `<js-install-6.4>/buildomatic`
3. Rename the file `default_master.properties`
4. Edit `default_master.properties` for your database and application server.

Database	Sample Property Values
MySQL	appServerType=tomcat6 [tomcat7, jboss, glassfish2, glassfish3] appServerDir=C:\\Apache Software Foundation\\Tomcat 7 dbUsername=root dbPassword=password dbHost=localhost

3.7 Upgrading to JasperReports Server 6.4

Now that your buildomatic scripts are configured, you can complete the upgrade.

Make sure you've backed up your `jasperserver` database before proceeding.

Make sure you've backed up your old JasperReports Server WAR file before proceeding.

1. Stop your application server
2. Start your database server
3. Run the following commands:

Commands	Description
`cd <js-install-6.4>/buildomatic`	Change to buildomatic directory
`js-upgrade-newdb-ce.bat <path>\js-6.2-export.zip`	(Windows) Upgrade jasperserver war file, drop and recreate the database, import data file from previous version.
`./js-upgrade-newdb-ce.sh <path>/js-6.2-export.zip`	(Linux) Upgrade jasperserver war file, drop and recreate the database, import data file from previous version.

On MySQL, if you receive an error about packet size, see the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.

3.7.1 js-upgrade Test Mode

Use the `test` option to run the `js-upgrade` script in test mode. For example, on Windows, enter:

```
cd <js-install-6.4>/buildomatic
js-upgrade-newdb-ce.bat test <path>/js-6.2-export.zip
```

In test mode, the js-upgrade scripts check your default_master.properties settings and validate your application server location and its ability to connect to your database. Test mode can help you debug issues like an incorrect database password without altering your system.

3.7.2 Output Log Location

The js-upgrade script creates an output log that captures both standard and error output. If problems occur during script execution, or you just want to remember which options you chose, open the output log file located here:

`<js-install-6.4>/buildomatic/logs/js-upgrade-<date>-<number>.log`

3.7.3 Errors

If you encounter errors running the `js-upgrade` script, first look at the output log to see if you can spot the errors. For help, refer to the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*. The information in this appendix applies to both `js-upgrade` scripts and `js-install` scripts.

If you need to modify values in your `default_master.properties` file, you can simply edit the file. When you run the `js-upgrade` script again, it uses the new values.

3.8 Starting and Logging into JasperReports Server 6.4

Start your application server. Your database should already be running.

3.8.1 Clearing Your Browser Cache

Before you log in, make sure you and your end users clear the browser cache. JavaScript files, which enable the UI elements of JasperReports Server, are typically cached by the browser. Clear the cache to ensure that the newer files are used.

3.8.2 Logging into JasperReports Server

Log in using the following URL, user IDs, and passwords:

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

User ID	Password	Description
`jasperadmin`	`<your-password>`	Administrator for the default organization

Your JasperReports Server instance has now been upgraded to 6.4. If you have startup or login problems, refer to the Troubleshooting appendix of the *JasperReports Server Community Project Installation Guide*.

3.9 Additional Tasks to Complete the Upgrade

Perform these tasks with the application server shutdown.

3.9.1 Handling JasperReports Server Customizations

If you made modifications to the original JasperReports Server application, these configurations are typically found in the `WEB-INF/applicationContext-*.xml` set of files.

You'll need to manually copy configuration changes, like client-specific security classes or LDAP server configurations, from your previous environment and integrate them with your upgraded environment.

3.9.2 Clearing the Application Server Work Folder

Application servers have work folders where JasperReports Server files are compiled and cached and other objects are stored. When you update the WAR file, the buildomatic `deploy-webapp-ce` target should automatically clear the application server's `work` directory, but it's a good practice to double-check. A permission problem, or some other problem, could prevent the clearing of the work folder.

To clear the work folder in Tomcat:

1. Change directory to `<tomcat>/work`.
2. Delete all the files and folders in this directory.

3.9.3 Clearing the Application Server Temp Folder

JasperReports Server uses caching to speed operations within the application. Caching files are created and stored in the application server, usually in a `temp` folder. Clear this `temp` folder to avoid any post-upgrade conflicts. Typically, the `temp` folder used by an application server corresponds to the path referenced by the `java.io.tmpdir` Java system property. For Apache Tomcat the `temp` folder is `<tomcat>/temp`.

To clear the temp folder in Apache Tomcat:

1. Change directory to `<tomcat>/temp`
2. Delete all the files and folders in this directory

3.9.4 Clearing the Repository Cache Database Table

In the `jasperserver` database, compiled JasperReports Library resources are cached in the `JIRepositoryCache` table for increased efficiency at runtime. Because the JasperReports Library JAR is typically updated with each new release, old cached items can get out of date and cause errors at runtime. If you encounter errors that mention a JasperReports Library "local class incompatible," check your repository cache table. In summary, you can clear your `jasperserver` database cache table as part of this upgrade process whether or not there are errors.

To manually clear the repository cache database table, run a SQL command similar to one shown below:

```
update JIRepositoryCache set item_reference = null;
delete from JIRepositoryCache;
```

3.10 Old Manual Upgrade Steps

This section describes the older, manual upgrade steps used before we implemented the `js-upgrade` shell scripts in release 4.0. They're provided here mainly as a reference for internal use.

We recommend using the js-upgrade shell scripts described in the beginning of this chapter instead of these manual commands.

Commands	Description
`cd <js-install-6.4>/buildomatic`	

`js-ant drop-js-db`
`js-ant create-js-db`
`js-ant init-js-db-ce`
	Deletes and recreates your `jasperserver` db. Make sure your original database is backed up.

`js-ant import-minimal-ce`	
`js-ant import-upgrade` `-DimportFile="<path-and-filename>"`	The `-DimportFile` should point to the <path> and <filename> of the js-6.2-export.zip file you created earlier. On Windows, you must use double quotation marks (“) if your path or filename contains spaces. On Linux, you must use double quotation marks, escaped with a backslash (\”) in this case. Note: "import-upgrade" will import resources from the 6.2 instance in a "non-update" mode (so that core resources from 6.4 will stay unchanged). Additionally, the "update-core-users" option will be applied so that the superuser and jasperadmin users will have the same password as set in the 6.2 instance.
`js-ant import-sample-data-upgrade-ce`	(Optional) This step is optional; it loads the new sample data. The old sample data is overwritten, so you may need to redo certain changes such as configuring the sample data sources for your database.
`js-ant deploy-webapp-ce`	Deletes the existing older war file, deploys the new war file.

CHAPTER 4 UPGRADING JASPERSERVER 4.2.1 OR EARLIER

4.1 Upgrading from 4.2.1 or Earlier

If you're running JasperServer version 4.2.1, your upgrade requires two steps:

1. Upgrade from version 4.2.1 to version 6.3.
2. Upgrade from version 6.3 to version 6.4.

The steps for this upgrade are documented in the *JasperServer Installation Guide* for the 6.3 release. Download the JasperServer 6.3 WAR file distribution zip package to get the relevant files and documentation. The Installation Guide is in the docs folder.

If you're running a JasperServer version earlier than 3.7, first upgrade to 3.7, then to 6.3, then to 6.4.

4.2 Best Practices for Upgrading on Windows

The two methods for installing JasperReports Server are:

1. Installing with the Binary Installer and Bundled Components

   The binary installer is an executable that puts all the components in place to run JasperReports Server. For example, if you take the default installation choices, you'll get the Apache Tomcat application server, the PostgreSQL database and Java execution environment.

   But keep in mind that these components are specially configured to run a specific version of JasperReports Server. This applies to the Windows Start Menu items created to start and stop JasperReports Server.
2. Installing to Pre-existing Components

   When installing a “Production” instance of JasperReports Server, you may want to install the main components before you install JasperReports Server. This way you have more control over updating and upgrading components like the application server, database, and Java.

   Once you put these components in place, you have two options for installing JasperReports Server:

   a. Use the War File ZIP distribution (file name: TIB_js-jrs_cp_6.4.0_bin.zip)

      You'll install JasperReports Server to the existing components using the `js-install.bat` scripts. You'll create a default_master.properties file that specifies the location of the application server and database components.

   b. Use the Binary Installer, TIB_js-jrs_cp_6.4.0__installer-windows-x64.exe

      The installer will prompt you for the location of the application server and database components.

If you intend to upgrade your Windows installation with future releases of JasperReports Server, we recommend installing to pre-existing components. This will reduce any post-upgrade confusion caused by the Windows Start Menu showing the older version of JasperReports Server.

APPENDIX A PLANNING YOUR UPGRADE

Some of the new and enhanced features in JasperReports Server can affect your deployment, and you should plan your upgrade accordingly. Before upgrading make sure to:

- Review this information carefully and determine how the changes described affect your deployment.
- Back up your current JasperReports Server installation and repository.

The versions and their affected functionality are:

- Changes in 6.4 affect the Impala community connector.
- Changes in 6.2.1 affect the Impala community connector.
- Changes in 6.2 affect the default Ad Hoc templates.
- Changes in 6.1 affect themes.
- Changes in 6.0.1 affect Spring Security, including external authentication and customizations involving Spring Security.
- Changes in 6.0 affect custom data sources.
- Changes in 5.6 affect XML/A connections.

Changes are cumulative, so review all topics that affect you. For example, if you're upgrading from 5.6 to 6.1, you may be affected by changes in 5.6, 6.0, and 6.0.1.

This section describes only those changes that can significantly impact your existing deployment. For an overview of new features, improvements, and bug fixes see the release notes in the root directory of the distribution. For information on how to use the new features, see the *JasperReports Server User Guide* or the *JasperReports Server Administrator Guide*.

This chapter contains the following sections:

A.1 Changes in 6.4 That May Affect Your Upgrade

A.1.1 Removal of the Impala Connector

JasperReports Server provides new and updated drivers for various databases. In the 6.2 release, the previous connector for Impala that had been available on the Jaspersoft community website was replaced with the Simba JDBC driver (Cloudera-endorsed JDBC interface).

By default, the new release supports the new JDBC driver, and the old Impala connector cannot be used. You should update your Impala data sources to use the new driver. For more information, see the *JasperReports Server Community Project Administrator Guide*.

If you wish to continue using the Impala connector that was previously available from the community website, modify the install as described below.

1. Add the following files to the <js-install>/WEB-INF/lib directory:
    - hive-service-0.12.0-cdh5.1.3.jar
    - zookeeper-3.4.5-cdh5.1.3.jar
    - avro-1.7.5-cdh5.1.3.jar
    - commons-compress-1.4.1.jar
    - hadoop-core-1.2.1.jar
    - hive-ant-0.12.0-cdh5.1.3.jar
    - hive-common-0.12.0-cdh5.1.3.jar
    - hive-exec-0.12.0-cdh5.1.3.jar
    - hive-jdbc-0.12.0-cdh5.1.3.jar
    - jasperserver-hive-connector-bugfix-SNAPSHOT.jar
    - js-hive-datasource-1.2.1-cdh5.jar
    - paranamer-2.3.jar
    - parquet-hadoop-bundle-1.2.5-cdh5.1.3.jar
    - xz-1.0.jar
2. Delete the file applicationContext-HiveDatasource.xml from the <js-install>/WEB-INF directory:

If you do not add the files listed, data sources that use the old Impala connector will cause errors when running reports that rely on them.

A.2 Changes in 6.2.1 That May Affect Your Upgrade

A.2.1 Removal of the Impala Connector

JasperReports Server provides new and updated drivers for various databases. In this release, the previous connector for Impala that was available on the Jaspersoft community website is replaced with the Simba JDBC driver (Cloudera-endorsed JDBC interface).

By default, the new release supports the new JDBC driver, and the old Impala connector cannot be used. You should update your Impala data sources to use the new driver. For more information, see the *JasperReports Server Community Project Administrator Guide*.

If you wish to continue using the Impala connector from the community website, you must replace the following JAR files in the .../WEB-INF/lib directory:

Replace	With This
hive-service-0.13.1.jar	hive-service-0.12.0-cdh5.1.3.jar
zookeeper-3.4.6.jar	zookeeper-3.4.5-cdh5.1.3.jar

Unless you replace the files listed in the table, data sources that use the old Impala connector will cause errors when running reports that rely on them.

A.3 Changes in 6.1 That May Affect Your Upgrade

A.3.1 Changes to Themes

The look and feel of the JasperReports Server web interface has been redesigned to modernize the application's appearance. To accomplish this, markup and styles have been modified. As a result of these modifications, custom themes developed for the previous interface will need to be updated for the new interface.

The following table lists the changes made to the user interface and describes some of the steps necessary to update custom themes in overrides_custom.css. The main changes are in the banner, body, footer, and login page. The changes to the login page are extensive. Instead of attempting to update an existing login page, you should re-implement the login page in the new default theme.

For information on developing new themes, see the *JasperReports Server Community Project Administrator Guide* and the *JasperReports Server Ultimate Guide*.

Table A-1 Updating Themes in JasperReports Server 6.1

Element	Classname and Modifications	File	Notes
Banner	`.banner` Give custom value to `height`	containers.css	Default value: height: 32px
Body	`#frame` Set custom `top` and `bottom` values that position the body of the application between the banner and footer without overlap	containers.css	Default value: top: 32px bottom: 17px This value needs to be equal to or greater than the height of `.banner` The bottom position needs to be adjusted only if the height of the footer is changed

Element	Classname and Modifications	File	Notes
Banner Logo	`#logo` Give custom values to height and width that match the dimensions of your logo Adjust margins around the logo if needed	theme.css	Default values: height: 22px width: 176px margin-top: 6px margin-right: 4px margin-bottom: 0 margin-left: 8px
Banner Main Navigation	`.menu.primaryNav .wrap` Set height and line-height to 1px shorter than `.banner`	containers.css	height: 31px line-height: 31px
Banner Main Navigation Home icon	`.menu.primaryNav #main_home .wrap > .icon` Set height to be the same as .banner Set values for width and background-position to fit your image.	containers.css	height: 32px width: 14px background-position: 0 -164px background-position: 0 -163px (IE8-9)
Banner Main Navigation Item arrow icon	`.menu.primaryNav .node > .wrap > .icon` Set height to your desired value, with the maximum value being the same height measurement as the .banner element. Set background-position and width to a value that properly displays the default or your custom image.	containers.css	height: 32px background-position: left -79px width: 11px

Banner
Main Navigation
Item arrow icon
	`.menu.primaryNav .wrap.over`
	`.menu.primaryNav .wrap.pressed`
	Set background-position to a value that properly displays the default or your custom image.
		containers.css	background-position is not explicitly defined, the value is cascaded from `.menu.primaryNav .node > .wrap > .icon`
			This only needs to be adjusted if you want a different color disclosure indicator for the pressed and over states of the main menu links

Element	Classname and Modifications	File	Notes
Banner Search container	`#globalSearch.searchLockup` Set margin-top to desired value that will vertically center it within the banner.	controls.css	margin-top: 5px
Banner Metadata	`#metalinks li` Set line-height to the desired value that will vertically center it within the banner.	themes.css	line-height: 20px
Footer	`#frameFooter` Set height if you want it to be anything other than the default value.	containers.css	height: 17px
Login page	Re-implement in new theme.		

A.4 Changes in 6.0.1 That May Affect Your Upgrade

JasperReports Server uses the Spring Security framework to implement security throughout the product. In JasperReports Server 6.0.1, the Spring Security framework was updated from Spring Security 2.0.x to 3.2.5. For many users, this upgrade will have no impact. However, you may need to make some changes if you have implemented the following:

- External authentication – If you have implemented external authentication or single sign-on in your server implementation, you need to update your implementation:
  - If you implemented external authentication using one of the sample files included in the project, you need to reimplement your changes in the updated sample files in JasperReports Server 6.0.1.
  - If you implemented a custom external authentication solution, you need to migrate your solution to the new framework.
- Customizations – If you have customized the server using Spring Security classes, you need to migrate your solution to the new framework.

A.4.1 Migrating External Authentication Sample Files

If you have implemented external authentication using one of the sample-applicationContext-<customName>.xml files in the <js-install>/samples/externalAuth-sample-config directory, do the following to migrate your changes to JasperReports Server 6.0.1:

1. Before upgrading, back up your applicationContext-<customName>.xml file (for example, applicationContext-externalAuth-LDAP.xml), located in the <js-webapp>/WEB-INF directory of your previous version of JasperReports Server.
2. Update your server installation to JasperReports Server 6.0.1, as described in the *JasperReports Server Community Project Upgrade Guide*.

As of JasperReports Server 6.0.1, you can customize the default admin users created when external authentication creates a new organization. Optionally you can also encrypt the admin's password in the configuration files. If you want to encrypt the default password, you need to set this up before installation or upgrade. See the *JasperReports Server External Authentication Cookbook* and the *JasperReports Server Security Guide* for more information.

3. In the new installation, locate the sample file that corresponds to the file you implemented previously. For example, if you implemented applicationContext-externalAuth-LDAP.xml, locate <js-install-6.0.1>/samples/externalAuth-sample-config/sample-applicationContext-externalAuth-LDAP.xml.
4. Rename the JasperReports Server 6.0.1 sample file to remove the sample- prefix. For example, rename sample-applicationContext-externalAuth-LDAP.xml to applicationContext-externalAuth-LDAP.xml.
5. Configure the properties in the new sample file to match the properties in your existing sample file. To do this:
   a. Locate each bean you modified in the previous version.
   b. Find the same bean in the JasperReports Server 6.0.1 sample. The names of the beans are the same in each version.
   c. Copy or re-enter the properties you need for your server, taking care not to copy over class names or class packages.

Although the bean names are the same in the JasperReports Server 6.0.1 sample files, the name and package of the class in many bean definitions have changed. Make sure not to overwrite the new names with the old ones.

d. Save the JasperReports Server 6.0.1 sample file.

e. Rename the JasperReports Server 6.0.1 sample file to remove sample- prefix. For example, rename sample-applicationContext-externalAuth-LDAP.xml to applicationContext-externalAuth-LDAP.xml.

f. Place the modified file in the <js-webapp-6.0.1>/WEB-INF directory.

A.4.1.1 Wrapper Classes

To reduce the impact of future upgrades, we created wrapper classes for the Spring Security classes used in the external authentication sample files. **Table A-2, "Wrapper Classes in JasperReports Server 6.0.1," on page 33** shows the correspondence between Spring Security classes in earlier versions of JasperReports Server and the new wrapper classes in JasperReports Server 6.0.1.

A.4.2 Migrating Customizations

At a minimum, you need to change the names and paths of the Spring Security classes you reference in any customizations you've made to JasperReports Server. The Spring Security codebase was significantly restructured from 2.x to 3.x. Many classes were moved to new packages and some classes were renamed. **Table A-3, "Mapping of Spring Classes from 2.0.x to 3.2.5," on page 35** shows the mapping from 2.0.x to 3.2 for important Spring Security classes used in JasperReports Server. This table is for information only. It has not been verified with the Spring Security project and is not guaranteed to be correct. Additional information is included in the Spring Security 3.2.5 source code. You can also search the internet.

A.4.3 Changes to Security Classes in JasperReports Server 6.0.1

Table A-2 Wrapper Classes in JasperReports Server 6.0.1

Spring Security 2.0.x Class	3.x External Authentication Wrappers
org.springframework.security.ui.ExceptionTranslationFilter	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.JSExceptionTranslationFilter
org.springframework.security.providers.ProviderManager	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.JSProviderManager
org.springframework.security.ui.AuthenticationDetailsSourceImpl	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.JSAuthenticationDetailsSourceImpl
org.springframework.security.ui.cas.CasProcessingFilterEntryPoint	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.cas.JSCasAuthenticationEntryPoint
org.springframework.security.providers.cas.CasAuthenticationProvider	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.cas.JSCasAuthenticationProvider
org.jasig.cas.client.validation.Cas20ServiceTicketValidator	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.jasig.JSCas20ServiceTicketValidator
org.springframework.security.providers.cas.cache.EhCacheBasedTicketCache	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.JSEhCacheBasedTicketCache
org.springframework.cache.ehcache.EhCacheFactoryBean	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.JSEhCacheFactoryBean
org.springframework.security.userdetails.ldap.LdapUserDetailsService	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.ldap.JSLdapUserDetailsService

Spring Security 2.0.x Class	3.x External Authentication Wrappers
org.springframework.security.ldap.search.FilterBasedLdapUserSearch	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.ldap.JSFilterBasedLdapUserSearch
org.springframework.security.ldap.populator.DefaultLdapAuthoritiesPopulator	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.ldap.JSDefaultLdapAuthoritiesPopulator
org.springframework.security.ui.cas.ServiceProperties	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.cas.JSCASServiceProperties
org.springframework.security.providers.ldap.authenticator.BindAuthenticator	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.ldap.JSBindAuthenticator
org.springframework.security.ldap.populator.DefaultLdapAuthoritiesPopulator	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.ldap.JSDefaultLdapAuthoritiesPopulator
org.springframework.security.providers.ldap.LdapAuthenticationProvider	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.ldap.JSLdapAuthenticationProvider
org.springframework.http.client.SimpleClientHttpRequestFactory	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.JSSimpleClientHttpRequestFactory
org.springframework.jdbc.datasource.DriverManagerDataSource	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.jdbc.JSDriverManagerDataSource
org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvider	com.jaspersoft.jasperserver.api.security.externalAuth.wrappers.spring.preauth.JSPreAuthenticatedAuthenticationProvider

Table A-2 shows the 2.0.x Spring Security classes. Note that three of the Spring Security classes in the table have moved to different packages:

ExceptionTranslationFilter	moved to	org.springframework.security.web.access.ExceptionTranslationFilter
ProviderManager	moved to	org.springframework.security.authentication.ProviderManager
AuthenticationDetailsSourceImpl	moved to	org.springframework.security.authentication.AuthenticationDetailsSourceImpl

Table A-3 Mapping of Spring Classes from 2.0.x to 3.2.5

Spring Security 2.0.x Class	Spring Security 3.2 Class
org.springframework.security.context.SecurityContextHolder	org.springframework.security.core.context.SecurityContextHolder
org.springframework.security.Authentication	org.springframework.security.core.Authentication
org.springframework.security.util.FilterChainProxy	org.springframework.security.web.FilterChainProxy
org.springframework.security.providers.anonymous. AnonymousProcessingFilter	org.springframework.security.web.authentication. AnonymousAuthenticationFilter
org.springframework.security.ui.basicauth. BasicProcessingFilter	org.springframework.security.web.authentication.www. BasicAuthenticationFilter
org.springframework.security.ui.basicauth. BasicProcessingFilterEntryPoint	org.springframework.security.web.authentication.www. BasicAuthenticationEntryPoint
org.springframework.security.ui.ExceptionTranslationFilter	org.springframework.security.web.access.ExceptionTranslationFilter
org.springframework.security.ui.webapp. AuthenticationProcessingFilterEntryPoint	org.springframework.security.web.authentication. LoginUrlAuthenticationEntryPoint
org.springframework.security.ui.webapp. AuthenticationProcessingFilter	org.springframework.security.web.authentication. UsernamePasswordAuthenticationFilter

Spring Security 2.0.x Class	Spring Security 3.2 Class
org.springframework.security.context. HttpSessionContextIntegrationFilter	org.springframework.security.web.context. SecurityContextPersistenceFilter
org.springframework.security.vote.AffirmativeBased	org.springframework.security.access.vote.AffirmativeBased
org.springframework.security.vote.AuthenticatedVoter	org.springframework.security.access.vote.AuthenticatedVoter
org.springframework.security.intercept.web. FilterSecurityInterceptor	org.springframework.security.web.access.intercept. FilterSecurityInterceptor
com.jaspersoft.jasperserver.api.security.JSSwitchUserProcessingFilter	com.jaspersoft.jasperserver.war.common.JSSwitchUserProcessingFilter
org.springframework.security.acl.basic.AclObjectIdentity	org.springframework.security.acls.model.ObjectIdentity
org.springframework.security.GrantedAuthority	org.springframework.security.core.GrantedAuthority
org.springframework.security.userdetails.User	org.springframework.security.core.userdetails.User
org.springframework.security.core. AuthenticationServiceException	org.springframework.security.authentication. AuthenticationServiceException
org.springframework.security.AuthorizationServiceException	org.springframework.security.access.AuthorizationServiceException
org.springframework.security.providers. UsernamePasswordAuthenticationToken	org.springframework.security.authentication. UsernamePasswordAuthenticationToken
org.springframework.security.core.GrantedAuthorityImpl	org.springframework.security.core.authority.GrantedAuthorityImpl
org.springframework.security.vote.BasicAclEntryVoter	org.springframework.security.acls.AclEntryVoter
org.springframework.security.vote.AccessDecisionVoter	org.springframework.security.access.AccessDecisionVoter
org.springframework.security.acl.AclEntry	org.springframework.security.acls.model.Acl

Spring Security 2.0.x Class	Spring Security 3.2 Class
org.springframework.security.ConfigAttributeDefinition	Collection<org.springframework.security.access.ConfigAttribute>
org.springframework.security.ConfigAttribute	org.springframework.security.access.ConfigAttribute
org.springframework.security.AuthorizationServiceException	org.springframework.security.access.AuthorizationServiceException
org.springframework.security.afterinvocation.AfterInvocationProvider	org.springframework.security.access.AfterInvocationProvider
org.springframework.security.AccessDeniedException	org.springframework.security.access.AccessDeniedException
org.springframework.security.acl.AclManager	org.springframework.security.acls.model.AclService
org.springframework.security.concurrent.SessionRegistry	org.springframework.security.core.session.SessionRegistry
org.springframework.security.concurrent.SessionInformation	org.springframework.security.core.session.SessionInformation
org.springframework.security.SecurityConfig	org.springframework.security.access.SecurityConfig
org.springframework.security.AuthorizationServiceException	org.springframework.security.access.AuthorizationServiceException
org.springframework.security.providers.encoding.PasswordEncoder	org.springframework.security.authentication.encoding.PasswordEncoder
org.springframework.security.ui.WebAuthenticationDetails	org.springframework.security.web.authentication.WebAuthenticationDetails
org.springframework.security.providers.dao.DaoAuthenticationProvider	org.springframework.security.authentication.dao.DaoAuthenticationProvider
org.springframework.security.ui.switchuser. SwitchUserGrantedAuthority	org.springframework.security.web.authentication.switchuser. SwitchUserGrantedAuthority
org.springframework.security.vote.AbstractAclVoter	org.springframework.security.access.vote.AbstractAclVoter
org.springframework.security.providers.anonymous. AnonymousAuthenticationToken	org.springframework.security.authentication. AnonymousAuthenticationToken

Spring Security 2.0.x Class	Spring Security 3.2 Class
org.springframework.security.afterinvocation.AfterInvocationProvider	org.springframework.security.access.AfterInvocationProvider
org.springframework.security.AccessDeniedException	org.springframework.security.access.AccessDeniedException
org.springframework.security.core.AuthenticationManager	org.springframework.security.authentication.AuthenticationManager
org.springframework.security.ui.cas.CasProcessingFilter	org.springframework.security.cas.web.CasAuthenticationFilter
org.springframework.security.ui.rememberme.NullRememberMeServices	org.springframework.security.web.authentication.NullRememberMeServices
org.springframework.security.util.UrlUtils	org.springframework.security.web.util.UrlUtils
org.springframework.security.providers.AuthenticationProvider	org.springframework.security.authentication.AuthenticationProvider
org.springframework.security.userdetails.jdbc.JdbcDaoImpl	org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl
org.springframework.security.BadCredentialsException	org.springframework.security.authentication.BadCredentialsException
org.springframework.security.userdetails.memory.UserAttribute	org.springframework.security.core.userdetails.memory.UserAttribute
org.springframework.security.context.SecurityContext	org.springframework.security.core.context.SecurityContext
org.springframework.security.providers.TestingAuthenticationToken	org.springframework.security.authentication.TestingAuthenticationToken
org.springframework.security.userdetails.ldap.LdapUserDetails	org.springframework.security.ldap.userdetails.LdapUserDetails
org.springframework.security.vote.RoleVoter	org.springframework.security.access.vote.RoleVoter
org.springframework.security.intercept.method.aopalliance. MethodSecurityInterceptor	org.springframework.security.access.intercept.aopalliance. MethodSecurityInterceptor
org.springframework.security.vote.UnanimousBased	org.springframework.security.access.vote.UnanimousBased

Spring Security 2.0.x Class	Spring Security 3.2 Class
org.springframework.security.afterinvocation.AfterInvocationProviderManager	org.springframework.security.access.intercept.AfterInvocationProviderManager
org.springframework.security.providers.ProviderManager	org.springframework.security.authentication.ProviderManager
org.springframework.security.ui.AuthenticationDetailsSourceImpl	org.springframework.security.authentication.AuthenticationDetailsSourceImpl
org.springframework.security.afterinvocation. BasicAclEntryAfterInvocationCollectionFilteringProvider	org.springframework.security.acls.afterinvocation .AclEntryAfterInvocationCollectionFilteringProvider
org.springframework.security.userdetails.memory.InMemoryDaoImpl	org.springframework.security.core.userdetails.memory.InMemoryDaoImpl
org.springframework.security.event.authentication.LoggerListener	org.springframework.security.authentication.event.LoggerListener
org.springframework.security.wrapper. SecurityContextHolderAwareRequestFilter	org.springframework.security.web.servletapi. SecurityContextHolderAwareRequestFilter
org.springframework.security.afterinvocation. BasicAclEntryAfterInvocationProvider	org.springframework.security.acls.afterinvocation. AclEntryAfterInvocationProvider
org.springframework.security.concurrent.SessionRegistryImpl	org.springframework.security.core.session.SessionRegistryImpl

A.5 Changes in 6.0 That May Affect Your Upgrade

A.5.1 Changes to Custom Data Sources

Changes to the Spring configuration in JasperReports Server 6.0 can affect custom data sources. This change specifically affects beans which implement the `ReportDataSourceServiceFactory` interface .

Prior to JasperReports Server 6.0, if code in JasperReports Server accessed a bean of this type, it got the actual instance of the Spring bean as configured in the Spring XML file, and it could be cast to the concrete class. In JasperReports Server 6.0 and later, beans of this type are intercepted and other code sees a dynamic proxy instead of the actual bean. Since proxies can only represent interfaces, existing code that tries to cast the bean to a concrete class will break.

Casting is usually done to get access to methods on a more specific class or interface. You can update your custom data sources as follows:

- To access methods on an existing interface, either do a cast to that interface, or inject the property using the existing interface so no cast is needed.
- To access methods *not* on an existing interface, create an interface with the methods you need and have the target object implement that interface.

A.6 Changes in 5.6 That May Affect Your Upgrade

The following changes in 5.6 and later can significantly affect your deployment:

- Changes to OLAP engine: Due to change between versions of the OLAP engine, if you use Jaspersoft OLAP's XML/A functionality to connect to a remote JasperReports Server's XML/A sources, you must take additional steps to complete your upgrade to 5.6.

A.6.1 Changes to OLAP Engine

If you use Jaspersoft OLAP's XML/A functionality to connect to a remote JasperReports Server's XML/A sources, you must take additional steps to complete your upgrade to 5.6. This is because of a change between versions of the OLAP engine.

Once the new version of JasperReports Server is installed and running, locate all the XML/A connections that point to a remote JasperReports Server instance. Then edit the DataSource field to specify `JRS` as the DataSource portion of its value.

For example, in previous versions, the Foodmart XML/A connection specified:

```
Provider=Mondrian;DataSource=Foodmart;
```

During upgrade, this connection must be changed to:

```
Provider=Mondrian;DataSource=JR
```

Note that for 5.6, the trailing semicolon should be removed (the older 5.5 syntax includes a semicolon at the end).

For more information about creating and editing XML/A connections, refer to the *Jaspersoft OLAP User Guide*.

One reason you might have XML/A connections to remote instances of JasperReports Server is to create a load-balanced Jaspersoft OLAP environment. For more information, refer to the *Jaspersoft OLAP Ultimate Guide*.

APPENDIX B WORKING WITH JDBC DRIVERS

This section describes how to set up your installation to use a driver other than the default driver.

B.1 Open Source JDBC Drivers

For open source JDBC drivers, buildomatic is set up to use a single default driver. If you want to use a driver other than the default driver, you can modify the buildomatic property files that determine the default JDBC driver.

The buildomatic JDBC driver property files are set up to point to a specific driver jar. This allows for multiple driver jar files in the same `buildomatic/conf_source/db/<dbType>/jdbc` folder. During the installation procedure only the default driver jar is copied to your application server.

If you want to use a newer JDBC driver version or a different JDBC driver, you can modify the buildomatic properties seen in your `default_master.properties` file.

B.1.1 PostgreSQL Example

The buildomatic/conf_source/db/postgresql/jdbc folder contains these driver files:

```
postgresql-9.2-1002.jdbc3.jar
postgresql-9.2-1002.jdbc4.jar
```

If, for instance, you want to change the default driver used by PostgreSQL from type jdbc4 to jdbc3, edit your default_master.properties file:

Overlay upgrade: `<overlay-folder>/buildomatic/default_master.properties`

Other upgrade: `<js-install>/buildomatic/default_master.properties`

Uncomment and change:

```
# maven.jdbc.version=9.2-1002.jdbc4
```

To:

```
maven.jdbc.version=9.2-1002.jdbc3
```

When you next run a buildomatic command, such as deploy-webapp-ce, the jdbc3 driver will be copied to your application server.

B.1.2 MySQL Example

The `buildomatic/conf_source/db/mysql/jdbc` folder contains this driver file:

```
mariadb-java-client-1.1.2.jar
```

If, for instance, you want to use a JDBC driver built and distributed by the MySQL project, such as `mysql-connector-java-5.1.30-bin.jar`, you first need to download the driver from the MySQL Connector/J download location:

```
https://dev.mysql.com/downloads/connector/j/
```

Next, change your buildomatic configuration properties to point to this new driver.

Edit your default_master.properties file:

Overlay upgrade:	`<overlay-folder>/buildomatic/default_master.properties`
Other upgrade:	`<js-install>/buildomatic/default_master.properties`

Uncomment and change:

```
# jdbcDriverClass=com.mysql.jdbc.Driver
# maven.jdbc.groupId=mysql
# maven.jdbc.artifactId=mysql-connector-java
# maven.jdbc.version=5.1.30-bin
```

To:

```
jdbcDriverClass=com.mysql.jdbc.Driver
maven.jdbc.groupId=mysql
maven.jdbc.artifactId=mysql-connector-java
maven.jdbc.version=5.1.30-bin
```