



JasperReports® Server

Installation

Version 9.0.0 | January 2024



Contents

Contents	2
Introduction	10
Conventions	12
JasperReports Server Distributions	12
Release Notes	14
System Requirements	14
Support for Internationalization	15
Installation Types	15
Additional Buildomatic Configuration for Split Installation	16
Enabling Java Naming and Directory Interface (JNDI) Security	19
Additional Buildomatic Configuration for JNDI Security Installation	19
Create Read-only Users	21
Websphere Installation for Enabling JNDI Security	23
Weblogic Installation for JNDI Security	25
Enabling JNDI Security post Installing JasperReports Server	27
Running the Binary Installer for Evaluation	29
Installation Requirements	30
Pre-installation Tasks	30
Installation Account	31
Choosing Installer Components	32
Bundled Components	32
Preparing Existing Components	32
Selecting a Tomcat Configuration	33
Selecting a PostgreSQL Configuration	34
Selecting a Chrome/Chromium Configuration	37

Choosing Sample Data	39
Installation	39
Installing in GUI Mode	39
Installing Using the Command Line	42
Post-Installation Tasks	42
Updates Made by the Installer During Installation	43
Installer Log File	44
Setting your Java JVM Options	44
Installing a New License File	44
License File for Existing Tomcat as Windows Service	45
Configuration when Web Server is Run as Root on Linux	46
Starting and Stopping the Server	46
Start/Stop Menu — Windows	47
Start/Stop Scripts — Linux	49
Start/Stop Apps — Mac OSX	50
Logging into JasperReports Server	52
Uninstalling the Server	54
Windows	54
Linux	54
Mac OSX	54
Uninstall Survey	55
Installing the WAR File for Production	56
WAR File Distribution	57
Applications Supported by the WAR File Distribution	59
Database and Application Server Support	59
Operating System Support for Bash Shell	60
Installing the WAR File Using js-install Scripts	60
Installing Chrome/Chromium	66
Additional Steps for Using DB2 and js-install Scripts	66
Additional Steps for Using JBoss EAP, JBoss Web Server or Wildfly	67
JBoss EAP/Wildfly Installation	68

JBoss Web Server Installation	69
Additional Steps for Using the IBM JDK	69
Starting the Server	70
Logging into the Server	71
JasperReports Server Heartbeat	72
Log Files	72
Managing Log Settings	73
Troubleshooting Your Server Configuration	73
Startup Problems	73
Error Running a Report	73
Error Running js-install Scripts (js-install.bat/sh)	73
Problem Connecting to a Cloud Database Instance	75
Error Exporting Reports, Ad Hoc Views, and Dashboards	75
Installing the WAR File Manually	76
JVM Options, License Setup, Working with JDBC Drivers	80
Setting JVM Options for Application Servers	80
Tomcat and JBoss JVM Options	81
Changing JVM Options for Tomcat as a Windows Service	85
Changing JVM Options for Bundled Tomcat on Linux	86
Setting Up the JasperReports Server License	86
Default License Configuration for All Application Servers	87
User-Defined License Location	88
Working With JDBC Drivers	90
Open Source JDBC Drivers	90
Installing Database Vendor JDBC Drivers	92
Working with Oracle RAC	94
Application Server Copy-to Locations	96
Locating and Changing Buildomatic Configuration Files	96
Regenerating Buildomatic Settings	96
Locating Buildomatic-Generated Property Files	97
Buildomatic Location for JasperReports Server WAR File	97

Buildomatic Location for SQL Scripts	98
Buildomatic Location for Database Creation Scripts	98
Buildomatic Location for Sample Data Catalog ZIP Files	99
Hibernate Properties Settings	99
Database Connection Configuration Files	100
Configuring Report Scheduling	101
Mail Server Configuration Settings	102
Database Settings for the Quartz Driver Delegate Class	106
Settings for the Report Scheduler Web URI	107
Settings for the Quartz Table Prefix	108
Settings for Import-Export	108
Setting Properties in the default_master.properties File	109
Updating XML/A Connection Definitions	112
Installing the WAR File for WebSphere	114
Procedure for Installing and Deploying the WAR File in WebSphere	114
Installing WebSphere and a Database	115
Preparing Server Files	115
Configuring CSRFGuard, Hibernate, Quartz, and WebSphere Settings	117
Configuring a JDBC Provider in WebSphere	122
Deploying the WAR File in WebSphere	131
Setting JVM Options	132
Configuring Other Database Connections	134
Defining a JNDI Name and Sample Data Sources for MySQL	134
Defining a JNDI Name and Sample Data Sources for DB2	141
Defining a JNDI Name and Sample Data Sources for Oracle	150
Defining a JNDI Name and Sample Data Sources for SQL Server	157
Starting and Restarting JasperReports Server	161
Logging into the Server	161
Configuring Report Scheduling	162
Additional Fix for Scheduled Report with JNDI Data Source	163
Additional Change for Mail Server Authentication	163

Updating XML/A Connection Definitions (Optional)	164
Troubleshooting your Configuration	164
Startup Problems	164
Error Running Report	164
Filter Error Using MySQL	165
Error Creating Internationalized Name	165
Xerces Error	166
OLAP View Fails with Exception	166
Installing the WAR File for WebLogic	168
Procedure for Installing the WAR File for WebLogic	169
Setting Java Properties	180
Configuring Other Database Connections	181
Configuring Databases Drivers	181
Starting the Server	187
Logging into the Server	187
Configuring Report Scheduling	188
Restarting the Server	188
Updating XML/A Connection Definitions (Optional)	189
Troubleshooting Your JasperReports Server Configuration	189
Startup Problems	189
Error Running Report	189
Managing Keystore Files	189
Scalable Query Engine	192
Overview	192
Architecture	195
Filter	195
Proxy Servlet	196
Load Balancer	196
Worker Pods	196
Redis Cache	197

Autoscaler	198
Fluentd Logging	198
Downloading the Software	198
Prerequisites	199
Downloading From Git	199
Docker Configuration	200
Configuring the Docker Images	201
Building the Docker Images	202
Deploying with Docker Alone (Optional)	203
Deploying with Kubernetes	203
Creating a Secret	204
Configuring the Helm Chart	205
Specifying a JNDI Data Source	215
Deploying to Kubernetes	217
Configuring JasperReports Server	218
Logging and Debugging	220
Aggregate to File	220
Elasticsearch Kibana (ELK)	221
Troubleshooting	222
Binary Installer Freezes	222
Installer Log Files	222
Installer DebugTrace Mode	223
Error Running Buildomatic Scripts	223
Missing Java JDK	224
Forgot to Copy the File ant-contrib.jar	224
Failure with '\$' Character in Passwords in Buildomatic Scripts	225
Older Apache Ant Version	225
Unable to Edit Files on Windows 10	225
Bash Shell for Solaris, IBM AIX, HP UX and FreeBSD	226
Linux Issues	227
File Issues with Extended Character Sets on Linux	227

Linux Installer Issue with Unknown Host Error	227
Installation Error with Windows Path	228
Mac OSX Issues	229
Problem Starting JasperReports Server on Mac	229
Database-related Problems	230
Database Privileges Required By JasperReports Server	230
Database Connectivity Errors	231
Case-sensitive Collation in SQL Server	233
Configuring the Oracle or SQL Server Driver for NTLM Authentication	233
Maximum Packet Size in MySQL	234
Case Sensitivity for Table and Column Names	235
PostgreSQL: Job Scheduling Error	235
Performance Issues with Oracle JDBC Queries	236
Using an Oracle Service Name	236
Error Running a Scheduled Report	236
Error Running a Report	237
Save Error with DB2 Database	237
BeanDefinitionStoreException with DB2 Driver	238
JDBC Driver Loading Error on Import/Export from WebLogic or WebSphere	239
Application Server-related Problems	239
Memory Issues Running Under Tomcat	239
Java Out of Memory Error	239
JVM Crash	240
Configuration File Locations	241
Tomcat Installed Using apt-get/yum	241
JBoss Modifications	242
WebSphere Modifications	245
WebLogic Modifications	246
Disabling User Session Persistence in Application Servers	246
License-related Errors	247
License Not Found Errors	247
License Not Found or License Corrupt Error with Tomcat as a Service	247

Problems Importing and Exporting Data from the Repository	248
Exporting a Repository That Contains UTF-8	248
Problems with Upgrade	249
Include Audit Events on Upgrade	249
Overlay Upgrade Permissions Error with Bundled Installation	249
Overlay Upgrade Domain Issue with MySQL and MariaDB JDBC Driver	250
Manually Creating the JasperReports Server Database	251
PostgreSQL	252
MySQL	253
Oracle	255
DB2	259
SQL Server	261
Jaspersoft Documentation and Support Services	264
Legal and Third-Party Notices	266

Introduction

JasperReports® Server builds on 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. The products utilize common metadata and provide shared services, such as security, a repository, and scheduling. The server exposes comprehensive public interfaces. This enables seamless integration with other applications and the capability to add custom functionality with ease.



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 are licensed to use, or to upgrade your license, contact Jaspersoft.

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

- Easily create 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 in PDF format on the [Product Documentation website](#). You can also access PDF and HTML versions of these guides 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.
- The [Jaspersoft Community site](#) covers topics for:
 - Developers
 - System administrators
 - Business users
 - Data integration users
- 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 <https://www.jaspersoft.com/support>.

JasperReports Server is a component of both a community project and commercial offerings. Each of these integrates the standard features such as security, scheduling, web services interface, and much more for running and sharing reports. Commercial editions provide additional features for hosting large BI deployments, including:

- Ad Hoc views and reports
- advanced charts
- dashboards
- domains
- auditing
- multi-organization architecture

This chapter contains the following sections:

- [Conventions](#)

- [JasperReports Server Distributions](#)
- [Release Notes](#)
- [System Requirements](#)
- [Support for Internationalization](#)
- [Installation Types](#)

Conventions

This document uses the following conventions when referring to file locations:

Convention	Description
<js-install>	<p>For binary installations, the directory where JasperReports Server will be installed by the binary installer. This directory contains directories for applications used by the installer, such as apache-tomcat and postgresql, as well as directories for JasperReports Server, such as samples.</p> <p>For manual installations, the directory where you unpack the WAR file distribution js-jrs_9.0.0_bin.zip.</p>
<java>	The directory where Java is installed.
<jboss>	The directory where JBoss is installed.
<postgresql>	The directory where PostgreSQL is installed. If you use our bundled instance of PostgreSQL, it's in the <js-install> directory.
<tomcat>	The directory where Apache Tomcat is installed. If you use our bundled instance of Tomcat, it's in <js-install> directory.

JasperReports Server Distributions

JasperReports Server is a web application that runs in an app server and uses an external database to store its repository. Depending on your use case, JasperReports Server is

available in two distributions: a binary installer for evaluation and a WAR file for production.

- The binary installer is an executable file that includes the Tomcat application server and PostgreSQL database so that it is fully self-contained. The installer software leads you through the options and installs Tomcat and PostgreSQL with default settings for JasperReports Server to use. You also have the option to install the sample data for working with tutorials, which is recommended.

The binary installer runs on desktops and laptops under Linux, MacOS, and Windows (64-bit), so you can quickly install an instance of JasperReports Server and explore all of its features. However, the app server and the database server are not configured for performance and not intended for use in production. For more information, see [Running the Binary Installer for Evaluation](#).

- The WAR (web archive) file is a zipped file containing only the JasperReports Server web app. To install the WAR file in production, you must first install and configure one of the supported app servers (for example Tomcat, JBoss, WebLogic, or Websphere) and one of the supported databases (most major relational databases). These servers can be in the cloud or on premises, shared with other web apps or dedicated to JasperReports Server (recommended), but they should be configured for performance under the expected load, reliability (uptime), security, and backups.

The WAR file distribution includes scripts that you edit to specify your app server and database. When you run these scripts, they create the necessary tables in your database and copy the files of the web app to your app server. You can also import an existing repository if you are upgrading from a previous version of JasperReports Server. For more information, see [Installing the WAR File for Production](#). For the list of supported app servers and databases, see the Jaspersoft Platform Support Guide for this release.



The binary installer is intended for evaluation purposes only. To install TIBCO JasperReports Server for enterprise production environments, use the stand-alone WAR file distribution, which is the official TIBCO JasperReports Server installer.

Both the binary installer and the WAR file are available from TIBCO Technical Support (<https://www.jaspersoft.com/support>).

Release Notes

Release notes are included with each distribution and with each new update to a distribution.

Not all applications are immediately supported when a new JasperReports Server version is released. For instance, some applications require additional testing beyond what is completed for the initial General Availability (GA) release. To find out exactly what applications are supported with a particular distribution refer to the release notes in that distribution.

System Requirements

The following table contains the minimum and recommended resources for a full installation that includes PostgreSQL and an application server. The values are based on our own testing. You may find that JasperReports Server can run on systems with fewer resources or slower systems than stated in the minimum resources column. At the same time, it's possible to run out of resources with the recommended configuration. The success of your deployment depends on the intended load of the system, the number of concurrent users, the data sets, and whether the databases are installed on the same system as the JasperReports Server.

Resource	Footprint	Minimum	Recommended
Disk	~1.5 Gigabytes	10GB free	40GB +
RAM		8GB	12GB +
Processor		2 core minimum	2.5GHz + multi-core Pentium for Windows, Mac, and Linux

For additional system requirements, including Java version, see the *JasperReports Server Supported Platform Datasheet*.

Support for Internationalization

JasperReports Server supports the full Unicode character set using UTF-8 encoding. It also depends on the underlying database and application server to support the UTF-8 character encoding. UTF-8 is configured by default in the bundled Tomcat and PostgreSQL software. If you use any other software, refer to the JasperReports Server Administrator Guide for instructions about configuring software to support UTF-8.

Installation Types

As of version 8.0, JasperReports Server supports the following installations:

- **Compact installation:** The Repository, Audit, Access, and Monitoring tables are created in a single repository database. This is the same configuration as previous versions.
- **Split installation:** Only the Repository tables are created in the repository database. The Audit, Access, and Monitoring tables are created in a separate audit database. For servers with high loads or performance needs, this speeds up repository access by storing diagnostic logs separately.

The default installation is the Compact installation.



The Access and Audit events are now processed asynchronously using a thread pool. The default number of threads is 15 and is configurable in the `applicationContext-events-logging.xml` file. If you want to switch back to the synchronous mode, comment out the following section:

```
<bean id="loggingEventsService"  
class="com.jaspersoft.jasperserver.api.logging.service.impl.LoggingFacade">  
  <property name="asyncExecutor" ref="asyncEventsExecutor"/>  
</bean>
```



The binary installer does not support split installation. For split installation, use the stand-alone WAR file distribution, which is the official JasperReports Server installer.

Additional Buildomatic Configuration for Split Installation

The `default_master.properties` file handles the configuration for the Split installation.

To configure the `default_master.properties` file for the Split installation:

- Edit the `default_master.properties` file to configure settings specific to your database and application server.

Look for the line **Uncomment below settings ONLY for split installation** and uncomment the settings listed in the following [table](#).

For example: To uncomment `# installType=split`, change it to `installType=split`.

The following [table](#) lists the settings you need to uncomment with sample values for each supported database.

Sample Values for the `default_master.properties` File for Split Installation

Database	Sample Property Values
PostgreSQL	<pre># installType=split # audit.dbHost=localhost # audit.dbUsername=postgres # audit.dbPassword=postgres # audit.dbPort=5432 # audit.dbName=jsaudit</pre>
MySQL	<pre># installType=split # audit.dbHost=localhost # audit.dbUsername=root # audit.dbPassword=password # audit.dbPort=3306 # audit.dbName=jsaudit</pre>
Oracle	<pre># installType=split</pre>

Database	Sample Property Values
	<pre># audit.dbHost=localhost # audit.dbUsername=jsaudit # audit.dbPassword=password # audit.dbPort=1521 # audit.dbName=jsaudit # audit.sysUsername=system # audit.sysPassword=password # audit.sid=ORCL If you are using an Oracle service name instead of an SID: # audit.serviceName=: uncomment and add your service name. # audit.AdditionalAdminProperties=;SysLoginRole=SYSDBA;User=sys;Password=password</pre>
DB2	<pre># installType=split # audit.dbHost=localhost # audit.dbUsername=db2admin # audit.dbPassword=password # audit.dbPort=50000 # audit.dbName=JSAUDIT For DB2, the audit.dbName value must be in uppercase.</pre>
SQL Server	<pre># installType=split # audit.dbHost=localhost # audit.dbUsername=sa # audit.dbPassword=sa # audit.dbPort=1433 # audit.dbName=jsaudit</pre>



Note the following:

If the `installType=split` property is not configured, the installation will be compact.

The `audit.dbPort` property is specific to the database. You can change the values for other properties as required.

Each `sample_conf/<dbType>_master.properties` file contains the properties and appropriate sample values.

Enabling Java Naming and Directory Interface (JNDI) Security

Enabling JNDI security or restricted access provides access-control to data sources. With JNDI restricted access, read-only access is provided to data sources.

This chapter includes the following sections:

- [Additional Buildomatic Configuration for JNDI Security Installation](#)
- [Create Read-only Users](#)
- [WebSphere Installation for Enabling JNDI Security](#)
- [Weblogic Installation for JNDI Security](#)
- [Enabling JNDI Security post Installing JasperReports Server](#)

Additional Buildomatic Configuration for JNDI Security Installation

The `default_master.properties` file handles the configuration for the JNDI security installation.

To configure the `default_master.properties` file for the JNDI security installation:

- Edit the `default_master.properties` file to configure settings specific to your database and application server.



Look for the line **Disable Edit/Delete access to jasperserver and jasperserverAudit JNDI connections** and uncomment the settings listed in [Sample Values for the default_master.properties File for JNDI Restricted Access Installation](#).

For example: To uncomment `#jndi.restrictedAccess=true`, change it to `jndi.restrictedAccess=true`.

[Sample Values for the default_master.properties File for JNDI Restricted Access Installation](#) lists the settings that you need to uncomment with sample values for each supported database.

Sample Values for the default_master.properties File for JNDI Restricted Access Installation

Database	Sample Property Values
PostgreSQL	<pre># jndi.restrictedAccess=true # analytics.dbUsername=jasperuser # analytics.dbPassword=password # auditAnalytics.dbUsername=jasperuser # auditAnalytics.dbPassword=password</pre>
MySQL	<pre># jndi.restrictedAccess=true # analytics.dbUsername=jasperuser # analytics.dbPassword=password # auditAnalytics.dbUsername=jasperuser # auditAnalytics.dbPassword=password</pre>
Oracle	<pre># jndi.restrictedAccess=true # analytics.dbUsername=jasperuser # analytics.dbPassword=password # auditAnalytics.dbUsername=jasperuser # auditAnalytics.dbPassword=password</pre>
DB2	<pre># jndi.restrictedAccess=true # analytics.dbUsername=jasperuser # analytics.dbPassword=password # auditAnalytics.dbUsername=jasperuser # auditAnalytics.dbPassword=password</pre>
SQL Server	<pre># jndi.restrictedAccess=true # analytics.dbUsername=jasperuser # analytics.dbPassword=Pass@123 # auditAnalytics.dbUsername=jasperuser # auditAnalytics.dbPassword=Pass@123</pre>


Database	Sample Property Values
	 The password for SQL Server must be a combination of a special character, a number, an uppercase, and a lower case character.
	 Once the database is created, you must create read-only users. For details, refer to Create Read-only Users

Each `sample_conf/<dbType>_master.properties` file contains the properties and appropriate sample values.

You can enable JNDI restricted access while deploying JasperReports Server on Tomcat or JBoss EAP or Wildfly. For details, refer to the *JasperReports® Server Security Guide*.

Create Read-only Users

The following table lists the database and the respective steps to create read-only users:

Database	Steps to create read-only users
PostgreSQL	<p>Create user:</p> <pre>CREATE USER postgres WITH PASSWORD 'postgres';</pre> <p>Assign read-only permissions:</p> <pre>GRANT CONNECT ON DATABASE jasperserver TO jasperuser; GRANT USAGE ON SCHEMA public TO jasperuser; GRANT SELECT ON ALL TABLES IN SCHEMA public TO jasperuser;</pre>
	 The above steps are for the jasperserver database. If you use split installation follow the same steps for the jsaudit database too.
MySQL	Create user:

Database Steps to create read-only users

```
CREATE USER 'jasperuser' IDENTIFIED BY 'password';
```

Assign read-only permissions:

```
GRANT SELECT ON *.* TO 'jasperuser';
```

Or

```
GRANT SELECT, SHOW VIEW ON *.* TO ''jasperuser'' IDENTIFIED BY 'password';
```

Oracle Create user:

```
CREATE USER jasperuser IDENTIFIED BY password;
```

Assign read-only permissions:

```
GRANT CREATE SESSION TO jasperuser;
```

```
GRANT READ ANY TABLE TO jasperuser;
```

DB2 Create user:

```
sudo useradd jasperuser
```

```
sudo passwd jasperuser (enter password as 'password')
```

Assign read-only permissions:

```
login using db2inst1/password
```

```
#Connect to the database
```

```
db2 connect to JSPRSVR user db2inst1 using password
```

```
# Create the 'read-only' role if it does not exist
```

```
db2 "create role readonly"
```

```
# Retrieve the list of table names in the 'JSPRSVR' schema and save them to a file
```

```
db2 -x "SELECT tablename FROM syscat.tables WHERE tabschema = 'JSPRSVR'" > table_list.txt
```

```
# Grant 'SELECT' privilege for each table in the 'read-only' role
```

```
while IFS= read -r table; do
```

```
db2 "grant select on JSPRSVR.$table to role readonly"
```

```
done < table_list.txt
```

```
# Grant the 'read-only' role to the 'jasperserver' user
```

Database	Steps to create read-only users
	<pre>db2 "grant role readonly to user jasperuser" # Clean up - remove the temporary file rm table_list.txt</pre>
SQL Server	<p>Create user:</p> <pre>CREATE LOGIN jasperuser WITH PASSWORD ='Pass@123'; CREATE USER jasperuser FOR LOGIN jasperuser;</pre> <p>Assign read-only permissions:</p> <pre>ALTER ROLE db_datareader ADD MEMBER jasperuser;</pre>

Websphere Installation for Enabling JNDI Security

Procedure

1. Deploy VM on the Websphere application server.
2. Log in to the Websphere console https://<ip_address>:9043/ibm/console, using the credentials wasadmin and wasadmin.
3. Navigate to Resources > JDBC > Data Sources > Add New JNDI Data Sources. For details, refer to [Configuring a JDBC Provider in WebSphere](#).

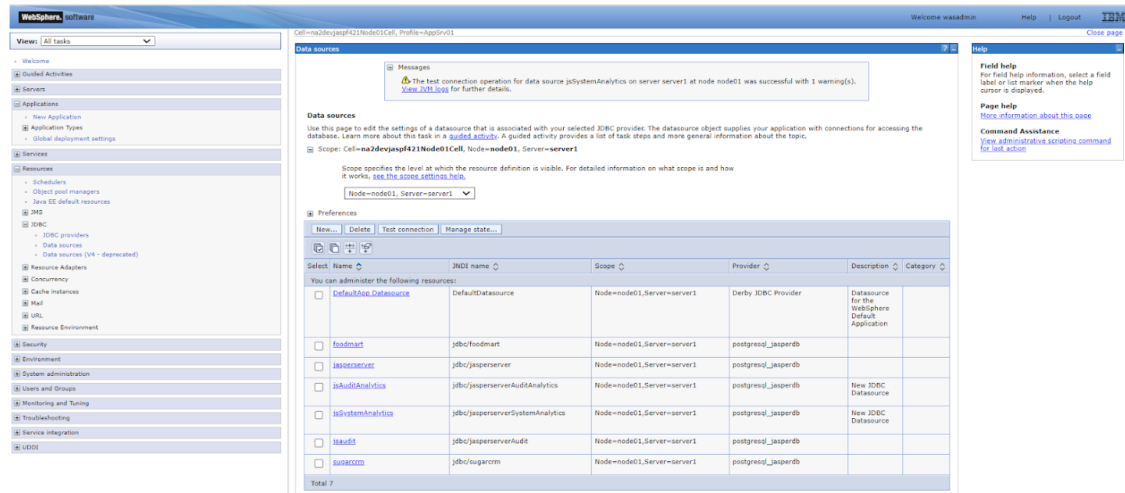


Figure 1: Add New JNDI Data Sources

- Restart the Websphere server.
- Navigate to Applications > Application Types > Websphere Enterprise Application > Select your application (for example, jasperserver-pro_war) > Stop > Start.

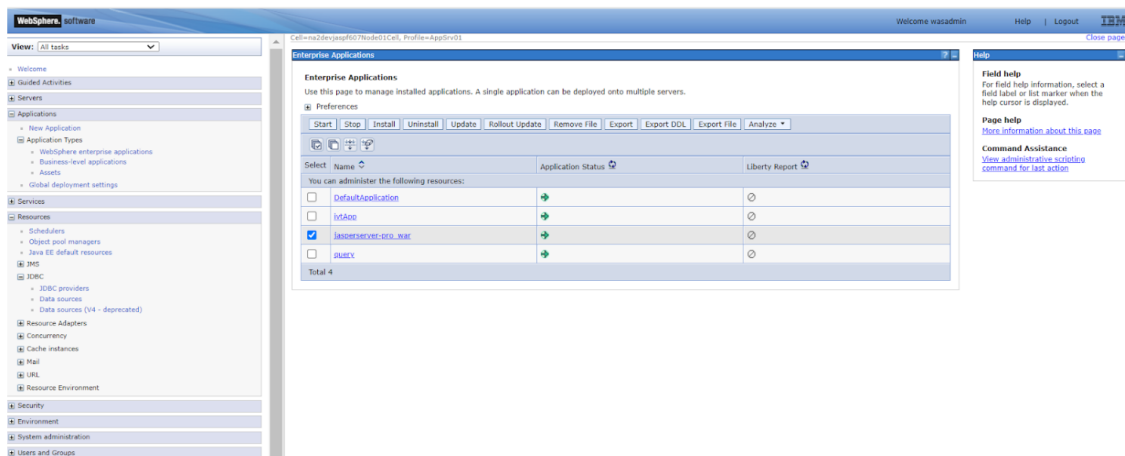


Figure 2: Restart Websphere Server

- Log in to JasperReports Server and check connection to all JNDI data sources.
- Set `hibernate.properties=true`.


```
cd
IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/
na2devjaspf607Node01Cell/jasperserver-pro_war.ear/jasperserver-
pro.war/WEBINF/classes/
vim hibernate.properties
```

8. Set `metadata.hibernate.jndi.restrictedAccess.enabled=true`.
9. Restart the application server again and go to JasperReports Server. The test connection should fail for `jasperserver` and `jasperserverAudit` data sources.

Weblogic Installation for JNDI Security

Procedure

1. Deploy VM on the Weblogic application server.
2. Log in to the Weblogic console `http://host:port/console` using the credentials `weblogic` and `just4eng`.
3. If the console is not accessible, then start Weblogic by using the following commands:

```
sudo chmod -R 777 Oracle
cd /opt/Oracle/Middleware/Oracle_Home/domains/jasper_domain/bin
sudo ./startWebLogic.sh
```
4. Navigate to Domain Structure > Services > Data Sources > Add AuditAnalyticsDataBase and JasperServerSystemDataBase. For details, refer to [Procedure for Installing the WAR File for WebLogic](#)

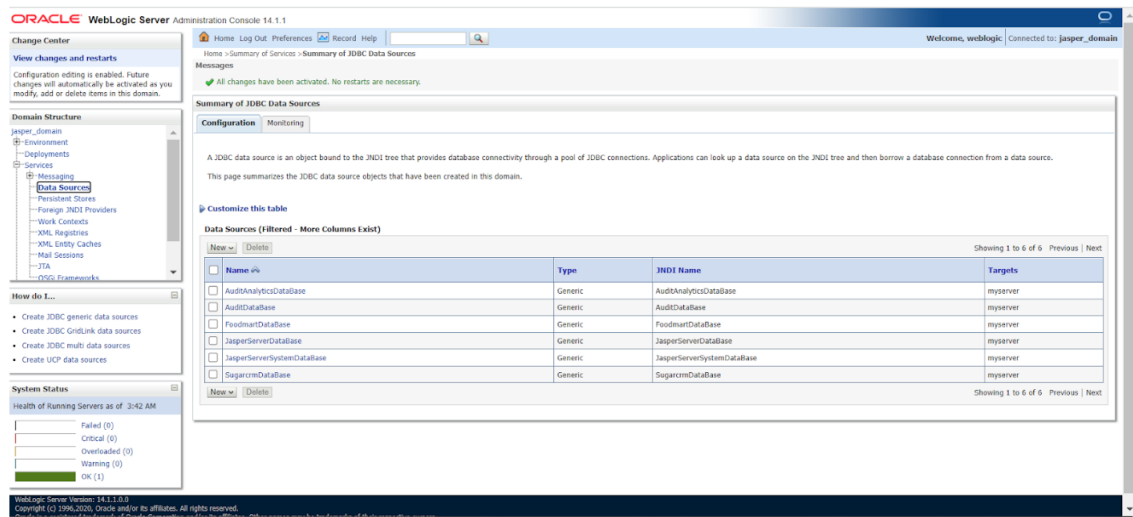


Figure 3: Add New JNDI Data Sources

5. Redeploy WAR file.
6. Navigate to Deployment.
7. Select jasperserver-pro file and click Update.

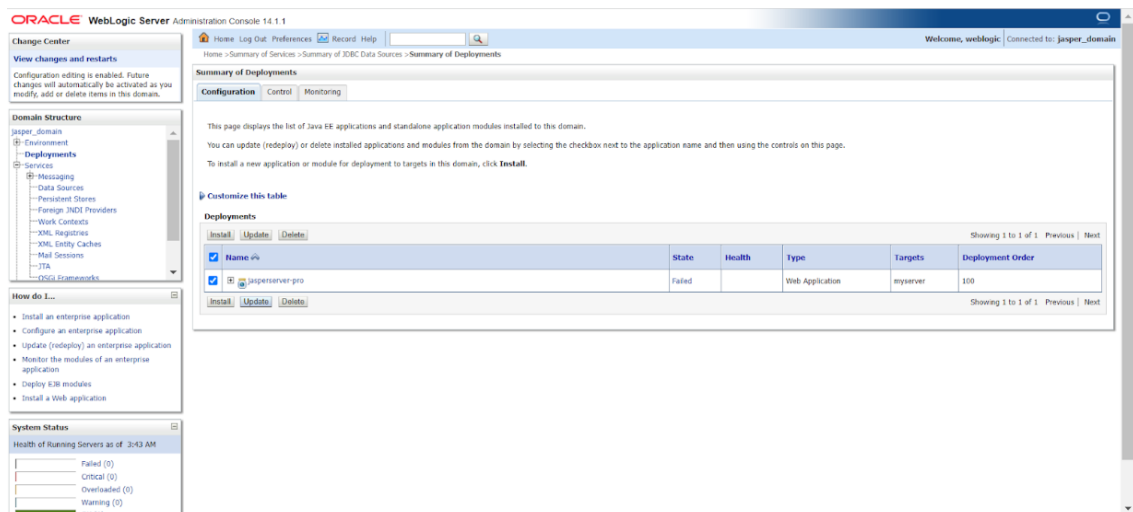


Figure 4: Redeploy WAR file

8. Log in to JasperReports Server and check the connection to all JNDI data sources.
9. Set `metadata.hibernate.jndi.restrictedAccess.enabled=true`.
10. Update the `hibernate.properties` in `jasperserver-pro.war` file.
11. Redeploy `jasperserver-pro.war` file.
12. Go to JasperReports Server. The test connection should fail for `jasperserver` and `jasperserverAudit` data sources.

Enabling JNDI Security post Installing JasperReports Server

Procedure

1. Follow the steps in [Create Read-only Users](#) to create read-only users.
2. Update the `jasperreports server>/WEB-INF/hibernate.properties` file and set `metadata.hibernate.jndi.restrictedAccess.enabled=true`.
3. Edit the `jasperreports server>/META-INF/context.xml` file.

Based on the values used in Step 1, change the username and password for the following two resources:

- `jdbc/jasperserverSystemAnalytics`
 - `jdbc/jasperserverAuditAnalytics`
4. Complete the changes and restart the application server.
 5. Create a JNDI data source by using either of the following two connections:
 - `jdbc/jasperserver`
 - `jdbc/jasperserverAudit`

A message stating *If your JNDI Data Connection has Edit/Delete access disabled, jasperserver and jasperserverAudit cannot be used for creating a new data source. Please select another JNDI connection.* is displayed.

6. Create a JNDI data source by using either of the following two connections:
 - `jdbc/jasperserverSystemAnalytics`

- jdbc/jasperserverAuditAnalytics

You should be able to create data source using these connections.

Running the Binary Installer for Evaluation

To install and evaluate a full working instance of JasperReports Server on a laptop or desktop, you can use the binary installer available for Windows, Linux, or Mac. The binary installer uses the Tomcat application server and the PostgreSQL database, installs the JasperReports Server web app, and optionally installs the sample data for working with tutorials.

Double-click the installer and accept the default installation type to create a standard installation. Select the custom installation to specify the Tomcat application server and/or PostgreSQL instance to use, among other options. The installer can also be run from the command line.



The binary installer is intended for evaluation purposes only. To install JasperReports® Server for enterprise production environments, use the stand-alone WAR file distribution, which is the official JasperReports® Server installer.

For more information, see [Installing the WAR File for Production](#).

This chapter includes the following sections:

- [Installation Requirements](#)
- [Choosing Installer Components](#)
- [Choosing Sample Data](#)
- [Installation](#)
- [Post-Installation Tasks](#)
- [Starting and Stopping the Server](#)
- [Logging into JasperReports Server](#)
- [Uninstalling the Server](#)

Installation Requirements

Before installing the product on your system, ensure that you can log in to the system with the appropriate permissions, and that your system meets the hardware and software requirements needed to install JasperReports Server. See the *JasperReports Server Supported Platform Datasheet* for more information.

Pre-installation Tasks

Pre-installation tasks include the tasks that you must complete before you start the installer, such as ensuring your system meets the installation requirements. If you want to use existing versions of any component software, you must also prepare them, as described in later sections.

Exclude JasperReports Server Installer File from Security Software Scans

If using anti-virus security software, you may need to ensure that the exception list includes the `js-jrs_8.x.x_win_x86_64.exe` installer file.

Download JasperReports Server Software

Download the JasperReports Server software package for your platform from the [Download section](#) of the Jaspersoft Community website or from the TIBCO Software Product Download Site (<https://edelivery.tibco.com/>). Extract the JasperReports Server archive file to a temporary directory on the machine on which you will run the installer.

The installers have the following file names:

- `js-jrs_9.0.0_win_x86_64.exe`
- `js-jrs_9.0.0_linux_x86_64.run`
- `js-jrs_9.0.0_macosx_x86_64.zip`

Installation Account

The privileges required to install the product differ for Windows and UNIX platforms. Ensure that you have the appropriate privileges to install on the target platform.

Microsoft Windows

You must have administrator privileges for the machine on which you want to install the software. Right-click the binary installer file and select “Run as administrator” from the context menu.



The Windows installer will get an error installing the PostgreSQL database if the Windows user does not have sufficient administrative privileges and if the installer is not started by right-clicking to use “Run as administrator”.

UNIX

In Linux, the installer is a `.run` file. You can run it from the command line or from a graphical environment. Any non-root user can perform the installation.

Mac

In Mac OSX, the installer file is `js-jrs_9.0.0_macosx_x86_64.zip`. After download, you should find the installer already unpacked in your `<user>/Downloads` folder.

When you double-click the installer application, you will see a security warning such as “<filename> cannot be opened because the developer cannot be verified.” The Mac installer app is signed by Jaspersoft®, but it has not been notarized because that requires separately signing all dynamic libraries and enabling the hardened runtime option. Click Cancel to exit the warning. There are two possible ways to run the installer:

- On older versions of Mac OSX, you can double-click again to get a different security dialog.
- On newer versions of Mac OSX, you hold control and click on the app, then select Open from the menu.

When run this way, there is another security warning such as "macOS cannot verify the developer of <filename>. Are you sure you want to open it?" Now there is an option to Open the app, so click that to launch the installer.

Choosing Installer Components

The installer is designed to get JasperReports Server up and running quickly. The server requires an application server and a database. The installation executable lets you choose whether to install bundled components or use versions of these already present on your system.

Bundled Components

The installer distribution bundles the following components:

Component	Description
JasperReports Server Application	WAR file and configuration support scripts.
Apache Tomcat	Web application container. You can use the bundled version or an existing version.
PostgreSQL Database	Database server. You can use the bundled version or an existing version.

Preparing Existing Components

You can use components that you have installed previously. Make sure that you are using supported versions of the components. For information about specific versions of third-party applications supported by the installer, refer to the JasperReports Server Supported Platform Datasheet.

- Tomcat application server: If you want to use an existing Tomcat, it must be on the local machine. See [Selecting a Tomcat Configuration](#) for more information.

- PostgreSQL database: If you want to use an existing PostgreSQL, it can be on a local or remote machine. If it's on a remote Linux machine, configure PostgreSQL to allow remote connections as described in [Enabling Connections to a Remote Host](#). See [Selecting a PostgreSQL Configuration](#) for more information.
- Chrome/Chromium browser: See [Selecting a Chrome/Chromium Configuration](#) for more information.



If you want to use an existing PostgreSQL database instance, the database must be running at install time. If you want to use an existing Apache Tomcat, the Tomcat instance must be stopped.

If you choose to install the bundled Tomcat and database, both are installed on the same host as the server.

Selecting a Tomcat Configuration

JasperReports Server requires an application server. The installer is configured to run with the Apache Tomcat server. You can choose to use a bundled Tomcat or an existing Tomcat.

Bundled Tomcat

If you select **I want to use the bundled Tomcat**, the installer puts an instance of Tomcat onto your system. If you are prompted for Tomcat's server port and shutdown port, you can accept the default values or enter alternate values. If a port is already in use, you receive an error. The installer looks for open Tomcat ports from 8080 up.

Existing Tomcat

If you want to use an existing Tomcat application server, **I want to use an existing Tomcat**. Later you are prompted for the location of Tomcat. Browse to the folder where you installed Tomcat. Make sure that the existing Tomcat is not running. When you are prompted for Tomcat's server port and shutdown port, you can accept the default values or enter alternate values.

Selecting a PostgreSQL Configuration

JasperReports Server requires a database. The installer is pre-configured to run with the PostgreSQL database. You may use any of the following two options:

- bundled PostgreSQL
- existing PostgreSQL

Choosing the Bundled PostgreSQL

If you install the bundled PostgreSQL, the installer puts PostgreSQL onto your system. The default PostgreSQL port is 5432. If port 5432 is in use, the installer prompts you to pick an alternate port.

The installer sets the PostgreSQL administrator password to **postgres** and creates a PostgreSQL database user with administrator privileges and the credentials **jasperdb/password**.

The following table summarizes the parameters set during installation of the bundled PostgreSQL:

Parameter	Default Value and Description
Binary Directory	The directory where the PostgreSQL and pgAdmin binaries are located.
Port	The port number PostgreSQL uses (default is 5432). Choose an alternate port if 5432 is in use.
IP or Host Name	The IP address or name of the machine where PostgreSQL is installed. The default value is 127.0.0.1.
PostgreSQL Administrative Password	Password of the database administrative user: password. The installer cannot handle special characters at the end of a password string. Incompatible characters include: & ; \$
Database User Name	The user name is hard coded. The default value is jasperdb. The installer creates this user to connect to the

Parameter	Default Value and Description
	JasperReports Server database.
Database User Password	The password is hard coded. The default value is password. The installer uses this password for the jasperdb user.
Additional notes for Linux	If your Linux installation does not have a locale setting that supports UTF-8 encoding, your bundled PostgreSQL instance is initialized using a temporary locale (<code>--locale=C</code>). This allows the PostgreSQL <code>initdb</code> to succeed with the desired UTF-8 database encoding.

Choosing an Existing PostgreSQL on a Local Host

If you choose to use an existing PostgreSQL database, then you are prompted for the location of PostgreSQL and the port to use. If you have an instance of PostgreSQL installed locally, accept the default, which is 127.0.0.1, the localhost. Accept the default location for the PostgreSQL `\bin` directory, or click Browse to select another location. You are also be prompted for the default administrative account password of the PostgreSQL administrative user. The database administrative user account name `postgres` is used by default. Enter the database administrative user password and click Enter.



If the installer displays with the following error message: `FATAL: password authentication failed for user postgres`, then try re-entering the administrative password for your PostgreSQL database.

The following table summarizes the parameters set during the installation of an existing PostgreSQL.

Defaults Used	Hard-coded Default Values Used or Created
PostgreSQL Administrative User Name	<code>postgres</code> - The default administrative database user.
jasperserver Database User	<code>jasperdb</code> - The installer creates this database user to

Defaults Used	Hard-coded Default Values Used or Created
Name	connect to the jasperserver database.
jasperserver Database User Password	password - The installer creates this password for the jasperdb database user.



To improve system security, Jaspersoft recommends that you change the default password for jasperdb as soon as possible. To change the jasperdb connection password in JasperReports Server, edit: `<js-install>/apache-tomcat/jasperserver-pro/META-INF/context.xml`. (And delete: `<js-install>/apache-tomcat/conf/Catalina/localhost/jasperserver-pro.xml`, if it exists.) Then make the same change in PostgreSQL using pgAdmin or psql.

Using an Existing PostgreSQL on a Remote Host

If you are installing on a remote instance of PostgreSQL, then you need to have the PostgreSQL client tools on your local machine. The client tools version should match the version of your remote PostgreSQL. You can check the version of the PostgreSQL instance by entering this command on the computer where it is installed:

```
psql --version
or
<path-to-postgresql-bin-folder>/psql --version
```

For instance: `C:/Jaspersoft/PostgreSQL/9.0/bin/psql --version`

To verify that, you can connect to the target remote PostgreSQL from the local installation machine

- Using your local PostgreSQL client tools, enter this command:

```
psql -U postgres -h <remote-host> -d postgres
or
<path-to-postgresql-bin-folder>/psql -U postgres -h <remote-host> -d postgres
```

You might also need to enable connections as described below.

Enabling Connections to a Remote Host

On most platforms, the default PostgreSQL installation does not allow remote connections for security reasons. You need to enable remote connections as described in this documentation:

- The PostgreSQL configuration documentation is on the PostgreSQL website.
- The `\docs` directory of your PostgreSQL installation.

To enable connections from the installation machine to the remote PostgreSQL server

1. Locate the following PostgreSQL host-based authentication (hba) configuration file on the remote PostgreSQL server instance:

Windows: `C:\Program Files\PostgreSQL\X.X\data\pg_hba.conf` (where X.X is the version number)

Linux: `/var/lib/pgsql/data/pg_hba.conf`

2. Add the IP address of your local JasperReports Server installation machine to this file. For example, to allow the local installation machine with address 192.168.12.10 to connect to the PostgreSQL server, add this entry to the `pg_hba.conf` file:

```
host all 192.168.12.10/32 trust
```

3. Allow TCP/IP connections to the remote PostgreSQL server instance by making the following change to the `postgresql.conf` file on the remote machine:

From: `listen_addresses = 'localhost'`

To: `listen_addresses = '*'`

4. Restart PostgreSQL.
5. Using your local PostgreSQL client tools, verify that you can connect to the target remote PostgreSQL from the local installation machine, as described in [Using an Existing PostgreSQL on a Remote Host](#).

Selecting a Chrome/Chromium Configuration

Chrome/Chromium executes JavaScript when generating graphical reports that are run in the background or scheduled. (When run directly in the web UI, the browser itself renders the graphics.) You have three options:

- Use an existing Chrome/Chromium
- Download Chrome/Chromium
- Install without Chrome/Chromium

Using an Existing Chrome/Chromium

If you choose to use an existing Chrome/Chromium executable, you'll be prompted for the location of Chrome/Chromium. If Chrome/Chromium is installed at the default location, installer will detect the path, or you can select another location.

The installer first checks for Chrome at the default location, if Chrome is not available, it checks for Chromium at the default location.



Auto-detection only works for Chrome/Chromium. A different path can be specified only for Chrome/Chromium in the installer. To use another browser, such as Edge, set the path in the `js.config.properties` file.

For information about configuring Chrome/Chromium or another browser in JasperReports Server, see the JasperReports® Server Administrator Guide.

Downloading Chrome/Chromium

If you do not have Chrome/Chromium installed, you can download Chrome or Chromium during installation. Installer provides the Chrome and Chromium download links.

For information about configuring Chrome/Chromium in JasperReports Server, see the JasperReports® Server Administrator Guide.

Installing without Chrome/Chromium

If you choose to continue the installation without Chrome/Chromium, reports and dashboards cannot be exported to PDF, DOCX, and other output formats.

Choosing Sample Data

During installation, you'll be prompted to install sample databases and sample reports. We provide these resources to help you evaluate the many features of JasperReports Server. This sample data includes:

- SugarCRM data that simulates 3 years of operations for a fictitious company that relies on the SugarCRM open source application.
- Foodmart data that simulates 3 years of operations for a fictitious company.
- JasperReports Server repository resources that use this data, such as reports, Ad Hoc Views, Domains, data sources, and input controls.

Our documentation provides tutorials that use this sample data. We strongly recommend that you install it.

Installation

Installing in GUI Mode

When you run the installer in GUI mode, the installer presents panels that you can use to choose the installation environment and customize your installation.

Steps for all installations

1. Run the installer executable. On Windows, make sure to right-click the binary installer file and select Run as administrator from the context menu.
 - On Microsoft Windows: `js-jrs_9.0.0_win_x86_64.exe`
 - On Unix: `js-jrs_9.0.0_linux_x86_64.run`
 - On Mac OS: `js-jrs_9.0.0_macosx_x86_64.zip`
2. On the welcome screen, click Next.
3. To accept the license agreement, click I accept the agreement then click Next.
4. Select the installation type that you want, then click Next.
 - Install All Components and Samples installs all components and samples.

- Custom Install lets you choose some pre-installed and some bundled components. Make sure you have installed supported versions of the components you want. If you are using a pre-installed Tomcat, make sure it is stopped. If you are using an existing PostgreSQL instance, it must be running during install. See the JasperReports Server Supported Platforms Datasheet for information about supported versions.
5. Select an installation path, then click Next. The default <js-install> directory depends on your operating system:

Windows:	C:\Jaspersoft\jasperreports-server-9.0.0
Linux:	<USER_HOME>/jasperreports-server-9.0.0
Linux (as root)	/opt/jasperreports-server-9.0.0
Mac OSX	/Applications/jasperreports-server-9.0.0



On Linux, choose a <js-install> path that is no more than 84 characters.

6. In the **Chromium folder** window, choose whether or not to use Chromium, then click **Next**.
 - Yes
 - No



JasperReports Server uses the Chromium browser engine when you export reports and dashboards to PDF and other formats. If you do not have Chrome/Chromium installed, you can download Chrome or Chromium using the links given on the **Chromium folder** window.

7. If you choose **Yes**, specify the Chromium binary you want to use. If Chrome/Chromium is installed at the default location, the installer detects the path, or click **Browse** to select another location.

Auto-detection only works for Chrome/Chromium. A different path can be specified only for Chrome/Chromium in the installer. To use another browser, such as Edge, set the path in the js.config.properties file.



For information about configuring Chrome/Chromium or another browser in JasperReports Server, see the JasperReports Server Administrator Guide.

8. If you choose **No**, a warning message is displayed. Click **Yes** to continue.



If you choose to continue the installation without Chrome/Chromium, reports and dashboards cannot be exported to PDF, DOCX, and other output formats.

9. If you plan to install JasperReports Server 9.0.0 on the same machine as an existing version 7.2.x or 7.5.x, you may encounter errors due to shared keystore files. Jaspersoft does not recommend running different versions on the same machine, but it is possible for evaluation purposes. You will need to follow additional steps after installation, as given by the link on the installer screen. To acknowledge this warning, click Next.

If you chose Install All Components and Samples, installation begins.

Additional steps for custom install

10. If you selected Custom Install, choose your components:

- Select the bundled or an existing Tomcat, then click Next.
- Select the bundled or an existing PostgreSQL database, then click Next. If you select an existing database, a warning popup appears. Click Yes to continue.

11. Enter your Tomcat ports, which may include server port, shutdown port, and AJP port. Then click Next.

12. Enter your database server port and click Next.

13. If you selected an existing Tomcat, specify your Tomcat directory and click Next.

14. If you selected an existing PostgreSQL, specify your PostgreSQL directory and click Next.

15. When you are prompted, choose whether or not to install sample databases and reports. Then click Next.

16. Click Next to begin the installation.

JasperReports® Server and any selected bundled components are installed on your system.

Finalizing the installation

17. Make your post-install selections on the final screen:

- Launch JasperReports Server Login Page - If you selected both bundled Tomcat and PostgreSQL, you see an option Launch JasperReports Server Login Page. If you are installing on Linux, then do not close the terminal window running the start script.

If you choose not to Launch JasperReports Server Now, the bundled components will not be started. If you have only one bundled component, it will not be started unless you use the Start/Stop menus or scripts.

- Opt-in for JasperServer Heartbeat - Sends anonymous system and version information to Jaspersoft using HTTPS.

18. Click Finish to complete the installation process and close the installer window.

Installing Using the Command Line

In Linux, the installer is a .run file; you can run it from the command line or from a graphical environment. To start the installer from the command line, open a bash shell, and enter the name of the installer file. For example:

```
.js-jrs_9.0.0_linux_x86_64.run
```

Whether you run the installer from the command line or in a graphical environment, you are prompted for the same information. If you are installing from the command line, use your keyboard to specify your answers. For example, with the license text, instead of clicking I accept the agreement, press Y, and press Enter.

Post-Installation Tasks



Installing JasperReports Server automatically generates encryption keys that reside on the file system. These keys are stored in a dedicated Jaspersoft keystore. Make sure that this keystore is properly secured and backed up, as described in the *JasperReports Server Security Guide*.

Updates Made by the Installer During Installation

This section lists the standard updates the installer makes to your local environment when you install. When the installation completes, you can check whether the updates, or corresponding changes, were successful.

Updates made to the application server

If you installed to an existing Tomcat, the installer attempted to make updates to the Tomcat environment, as shown in the following table.

File or Directory	Updates
Windows: bin/setclasspath.bat	Modifies JAVA_OPTS to add -Djs.license.directory.
Linux and Mac OSX: bin/setclasspath.sh	(Commercial installer only)
All platforms: lib	Adds JDBC drivers for databases to this directory.

Updates made to the PostgreSQL database

If you installed to an existing PostgreSQL database, the installer created new schemas and users in your database instance:

PostgreSQL Updates	Description
Database <code>jasperserver</code> created	This is the JasperReports Server repository database. This database holds all of system information, such as users, roles, data sources, and report definitions.
Database user <code>jasperdb</code> created	The JasperReports Server application uses this user to connect to the database.
Sample database <code>foodmart</code> created	(optional) Database created if install sample data option was chosen.
Sample database <code>sugarcrm</code> created	(optional) Database created if install sample data option was chosen.

Installer Log File

The installer creates a log during installation that records information as the installation progresses. If you encounter problems during installation, ensure that your system meets all prerequisites, and then check the installer log for potential problems.

The installer log file captures information such as:

- Environment details such as the user that invoked the installer, host name, operating system details, and so on.
- List of assemblies installed.
- Information related to the Ant scripts executed by the installer.

You can find the installer log at `<js-install>/installation.log`.

Setting your Java JVM Options

You need to set your Java JVM options. There are number of files where you can do this; refer to [Setting JVM Options for Application Servers](#).

Installing a New License File

By default, JasperReports Server is installed with an evaluation license that expires a number of days after installation. After the license expires, you can start the server, but you can't log in.

To obtain a commercial license, contact [Jaspersoft Technical Support](#) (<https://www.jaspersoft.com/support>) or your sales representative.

To upgrade the evaluation license to a commercial one, copy the commercial license file over the evaluation license file.

Application servers have work directories where JSP files are compiled and cached and other objects are stored. These directories can cause errors when upgrading a license. To avoid errors, clear the work directory before upgrading your license. For instance, if you're using Tomcat:

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

After changing to a commercial license, make sure you stop the server before replacing the license file:

1. Stop the server.
2. Replace the license named `jasperserver.license` in the deployed JasperReports Server root directory with the new license file.
The file name should be:
`jasperserver.license`
3. Restart the server.

By default, the license is in the `<js-install>` directory, but can be located elsewhere. You need to define the `-Djs.license.directory` Java Environment Variable in the Tomcat startup scripts to point to the license location. The name of the license file is `jasperserver.license`. Make sure the new license file has this name.

Restart JasperReports Server and log in to see if the license grants access. For information about license errors, see the troubleshooting section [License-related Errors](#).

For additional license configuration options, refer to [Setting Up the JasperReports Server License](#).

License File for Existing Tomcat as Windows Service

If you installed JasperReports Server into an existing Tomcat installation on a Windows system running as a Windows Service and the license file is not in the default location, because you didn't choose the default `<js-install>` installation directory, manually configure Tomcat to locate the license file.

Follow the steps below to examine and update the license location:

1. Open the Tomcat configuration tool by right-clicking the Tomcat icon in your quick-launch bar (usually in the lower-right corner of your desktop) or from the Windows 10 menu, expand `Start > Apache Tomcat`. Right-click `Configure Tomcat` and select `Run as administrator`.
2. Select the Java tab.
3. At the bottom of the Java Options field, enter the following option:
`-Djs.license.directory=<js-install>`
For example:
`-Djs.license.directory=C:\Jaspersoft\jasperreports-server-9.0.0`

4. Stop and restart the application server.

You should now be able to run JasperReports Server.

Configuration when Web Server is Run as Root on Linux

The export of reports, Ad Hoc views, and dashboards fails when Tomcat is run as root in the JasperReports Server installation on Linux. The Tomcat log file displays an error, for example:

```
2020-06-11T17:32:19,031 ERROR SecureExceptionHandlerImpl,http-nio-8080-exec-8:116 -
com.github.kklisura.cdt.launch.exceptions.ChromeProcessTimeoutException:
Failed while waiting for chrome to start: Timeout expired! Chrome
output: [0611/173218.897370:ERROR:zygote_host_impl_linux.cc(89)] Running
as root without --no-sandbox is not supported. See
https://crbug.com/638180.
```

To resolve this you need to set the following property in the `jasperreports.properties` file:

```
net.sf.jasperreports.chrome.argument.no-sandbox=true
```

After setting this property, restart JasperReports Server to enable it.

For information about configuring this property, see the *JasperReports Server Administrator Guide*.

Starting and Stopping the Server

- [Start/Stop Menu — Windows](#)
- [Start/Stop Scripts — Linux](#)
- [Start/Stop Apps — Mac OSX](#)

Start/Stop Menu — Windows

This section describes start and stop procedures that vary depending on whether you installed the bundled Tomcat and PostgreSQL or used your own Tomcat and PostgreSQL.

Start/Stop Menus — Bundled Tomcat and PostgreSQL

If you installed the bundled Tomcat and PostgreSQL, use the Windows Start menu to start and stop JasperReports Server.

- Click Start > JasperReports Server > Start Service.
- Click Start > JasperReports Server > Stop Service.

Additional Information about the Bundled Tomcat and PostgreSQL

The Windows installer installs PostgreSQL and Tomcat as Windows Services. You can manage these Services in the Windows Control Panel:

Control Panel > System and Security > Administrative Tools > Services

You can also start JasperReports Server from the Windows Start menu or by using the Desktop icon. You can shut down using the Desktop icon.

JasperReports Server Windows Service Names

PostgreSQL and Tomcat, installed as Windows Services, are listed in the Windows Services Panel as:

- jasperreportsPostgreSQL
- jasperreportsTomcat

Preventing JasperReports Server from starting up automatically

By default, the bundled services start automatically on a reboot, which also starts JasperReports Server. To change the startup mode for the services from automatic to manual:

- In the Windows Services Panel, select `jasperreportsTomcat`.
- Right-click the `jasperreportsTomcat` service, and select Properties.
- Change the Startup type drop-down setting from Automatic to Manual.
- Follow the same steps for `jasperreportsPostgreSQL` service.

To start JasperReports Server from the Windows Services Panel

1. Open the Windows Services Panel.
2. Select `jasperreportsPostgreSQL`, click Start.
3. Select `jasperreportsTomcat`, click Start.

To start JasperReports Server from the CMD Shell

1. Open a Windows CMD Shell.
2. Navigate to the root of the `<js-install>` folder (for example, `C:\Jaspersoft\jasperreports-server-9.0.0`)
 - a. To start JasperReports Server, run the following command:
`servicerun START`
 - b. To shutdown JasperReports Server, run the following command:
`servicerun STOP`

Running Processes

When JasperReports Server is running, the Windows Task Manager lists information about the processes running under the SYSTEM user name, for example:

- `postgres.exe`
- `tomcat9.exe`

Start/Stop Scripts — No Bundled Applications

During installation, if you chose to install one bundled and one existing Tomcat or PostgreSQL, you can use the Windows start/stop scripts to start and stop only the bundled one.

For example, if you have an existing Tomcat and you install the bundled PostgreSQL, the scripts and menus specified in the previous section would start and stop the PostgreSQL application. To start and stop the existing Tomcat, you would use the management scripts provided by the Tomcat application.



JasperReports Server requires database and application servers to be started in this order:

1. Database server.
 2. Application server.
-

Start/Stop Scripts — Linux

This section describes start and stop procedures that vary depending on whether you installed the bundled Tomcat and PostgreSQL or used your own Tomcat and PostgreSQL.

Manual Start/Stop

You typically start and stop JasperReports Server at the Linux command line. Run the following commands in a Linux shell.

- To start JasperReports Server, enter:

```
cd <js-install>  
./ctlscript.sh start
```
- To stop JasperReports Server, enter:

```
cd <js-install>  
./ctlscript.sh stop
```

To start and stop individual components:

```
cd <js-install>
./ctlscript.sh start|stop postgresql
./ctlscript.sh start|stop tomcat
```

If you enter just the `./ctlscript.sh` command, then you get the following:

```
./ctlscript.sh
usage: ./ctlscript.sh help
./ctlscript.sh (start|stop|restart|status)
./ctlscript.sh (start|stop|restart|status) postgresql
./ctlscript.sh (start|stop|restart|status) tomcat
help - this screen
start - start the service(s)
stop - stop the service(s)
restart - restart or start the service(s)
status - show the status of the service(s)
```

Auto Start/Stop with Bundled Tomcat and PostgreSQL

If you want JasperReports Server to start automatically when you reboot your Linux server, you need to install the JasperReports Server database and application server as services. If you have installed JasperReports Server using the binary installer with the bundled Tomcat and bundled PostgreSQL options, you'll find an example `jasperserver` service script in the following location:

```
<js-install>/scripts/linux/jasperserver
```

Edit this script and set permissions as described in the `<js-install>/scripts/linux/readme` file in the same location.

Once installed, these services start automatically when you reboot, which also starts JasperReports Server.

Start/Stop Apps — Mac OSX

After you complete the Mac OSX installation, you typically find JasperReports Server installed in the following location:

```
/Applications/jasperreports-server-9.0.0
```

When JasperReports Server is running, you can see the names of the Java and PostgreSQL processes in the Activity Monitor.

- To start JasperReports Server, locate this folder in Finder and double-click the following app:

`jasperServerStart.app`

- To stop JasperReports Server, locate this folder in Finder and double-click the following app:

`jasperServerStop.app`

The Mac lists the following information in the Activity Monitor:

- java
or
`org.apache.catalina.startup.Bootstrap`
- postgres

Start/Stop Apps — Mac Dock

Using Finder, move the following apps into the Mac Dock to start, stop, and login to JasperReports Server:

- `jasperServerStart.app`
- `jasperServerStop.app`
- `jasperServerLogin.app`

Start/Stop JasperReports Server — Mac Terminal Shell

To start and stop JasperReports Server using the Mac terminal shell

1. Open a Terminal shell (Finder > Go > Utilities > Terminal Icon).
2. Navigate to the <js-install> folder. For instance: `/Applications/jasperreports-server-9.0.0`
3. To start PostgreSQL, Tomcat, and JasperReports Server, enter:
`./ctlscript.sh start`

4. To shutdown PostgreSQL, Tomcat, and JasperReports Server, enter:

```
./ctlscript.sh stop
```

5. To start and stop individual components:

```
cd <js-install>  
./ctlscript.sh start|stop postgresql  
./ctlscript.sh start|stop tomcat
```

If you enter just the `./ctlscript.sh` command you will get the following:

```
./ctlscript.sh  
usage: ./ctlscript.sh help  
./ctlscript.sh (start|stop|restart|status)  
./ctlscript.sh (start|stop|restart|status) postgresql  
./ctlscript.sh (start|stop|restart|status) tomcat  
help - this screen  
start - start the service(s)  
stop - stop the service(s)  
restart - restart or start the service(s)  
status - show the status of the service(s)
```

Logging into JasperReports Server

To log into JasperReports Server on any operating system

1. Start JasperReports Server.
2. Open a supported browser: Firefox, Internet Explorer, Chrome, or Safari.
3. Log into JasperReports Server by entering the startup URL in your browser's address field. The URL depends upon your application server. If you installed the default, bundled Tomcat use:

```
http://<hostname>:8080/jasperserver-pro
```

- `<hostname>` is the name or IP address of the computer hosting JasperReports Server.
- 8080 is the default port number for the Apache Tomcat application server. If you used a different port when installing your application server, specify its port number instead of 8080.

The login page appears.

4. Log in using the following credentials:

User ID	Password	Description
superuser	superuser	System-wide administrator
jasperadmin	jasperadmin	Administrator for the default organization

If you installed the sample data, these additional sample end-users are also created. These users are non-administrative users with fewer system privileges.

User ID	Password	Description
joeuser	joeuser	Sample end-user
demo	demo	Sample end-user for the SuperMart Dashboard demonstration



When you complete the evaluation or testing of your JasperReports Server instance, change the administrator and superuser passwords (jasperadmin and superuser) and remove any sample end-users. Leaving the default passwords and end-users in place weakens the security of your installation.

To log into JasperReports Server on Windows

On Windows, you can launch the login page from the desktop of the JasperReports Server host computer by clicking Start > All Programs > JasperReports Server > JasperReports Server Login.

To log into JasperReports Server on Mac OSX

On Mac OSX, you can launch the login page by going to Finder and clicking the following script:

```
/Applications/<js-install>/jasperServerLogin
```

For example: /Applications/jasperreports-server-9.0.0/jasperServerLogin

To use the Dock to log into JasperReports Server:

From Finder, you can drag the /Applications/<js-install>/jasperServerLogin.app to the Dock to handle logging into JasperReports Server using your default system browser.

Uninstalling the Server

If you install JasperReports Server using the installer executable, you can uninstall it programmatically.

Windows

To uninstall JasperReports Server on Windows 10

Click Start > TIBCO > JasperReports Server > Uninstall.

Linux

On Linux, the <js-install> folder includes an executable that removes JasperReports Server from the host.

To uninstall JasperReports Server

1. From the command line, log in as any user with sufficient privileges.
2. Enter the following commands:

```
cd <js-install>
./uninstall
```
3. Respond Y or yes to the prompt that asks if you want to remove JasperReports Server from this computer.

Mac OSX

To use Finder to uninstall JasperReports Server

1. Navigate to the <js-install> folder.

2. Click the `uninstall.app` to launch the uninstaller.

Uninstall Survey

After running the uninstaller, you're prompted to take an uninstall survey from Jaspersoft. Survey answers are anonymous and help us improve our products. When you click Yes, the survey launches on the Jaspersoft website in a new browser window. Select all the reasons that led you to uninstall JasperReports Server. If none of the reasons apply, enter a short explanation. Thank you for your feedback.

Installing the WAR File for Production

For production environments, use the stand-alone WAR file distribution to install the JasperReports Server application. Download the WAR file distribution from [Jaspersoft Technical Support \(https://www.jaspersoft.com/support\)](https://www.jaspersoft.com/support) or contact your sales representative.

The WAR file distribution contains the JasperReports Server web archive file and the scripts to create and load the database. With the integration JasperReports Server and JasperReports Web Studio, two application WAR files, `jrws-jrio.war` and `jrws-repository.war`, are added to the WAR file distribution.



JasperReports Web Studio is the visual designer for creating and editing report templates for the reporting engine and the whole Jaspersoft family of products. It uses an open-source library to produce dynamic content and rich data visualizations. It comes as a web-based alternative to Jaspersoft Studio, the desktop application, which is the most complete and powerful designer for JasperReports templates.

Important Java Development Kit (JDK) 17 note: As of release 8.2, only Tomcat 9.0.x (the latest available) is supported by JasperReports Server on a system with JDK 17. An additional installation step is required on a system with JDK 17, which requires adding the `JAVA_OPTS` environment variable.

This chapter describes how to install the WAR file on the Apache Tomcat and JBossEAP/Wildfly application servers. For other application servers, see [Installing the WAR File for WebLogic](#) or [Installing the WAR File for WebSphere](#). For a list of supported JDK/JVMs, application servers, databases, operating systems, and browsers, refer to the TIBCO JasperReports® Server Supported Platform Datasheet.

This chapter contains the following sections:

- [WAR File Distribution](#)
- [Applications Supported by the WAR File Distribution](#)
- [Installing the WAR File Using js-install Scripts](#)
- [Additional Steps for Using DB2 and js-install Scripts](#)

- [Starting the Server](#)
- [Logging into the Server](#)
- [Troubleshooting Your Server Configuration](#)
- [Installing the WAR File Manually](#)

WAR File Distribution

The WAR file distribution comes in a file named `js-jrs_9.0.0_bin.zip` in compressed ZIP format.

The WAR file distribution includes `js-install` shell scripts (for Linux and Windows) that automate much of the installation using a single properties file. These scripts are:

- `js-install.bat`
- `js-install.sh`

The main contents of the WAR file binary distribution are shown in the following table.

Content Item	Description
JasperReports Server js-install Scripts	Found at <js-install>/buildomatic/js-install.bat and js-install.sh.
JasperReports Server Database Scripts	SQL scripts for each supported database.
JasperReports Server Documentation	Guides for end users and administrators.
JasperReports Server Extra Samples	Web Service example applications, sample reports, custom data source examples, and other sample files.
JasperReports Server Standard Sample Data	Sample data that highlights JasperReports Server features.
JasperReports Server WAR file archive	All JasperReports Server class files and dependent jars.
JasperReports IO for JasperReports Web Studio	JasperReports IO application is required for JasperReports Web Studio preview.
Repository for JasperReports Web Studio	Repository connector application is required for JasperReports Web Studio integration with JasperReports Server.

The application server should reside on the local machine, but the target database can be on a remote server. Using a remote PostgreSQL database on some Linux platforms requires a change to its configuration file, as described in [Enabling Connections to a Remote Host](#).

About Bundled Apache Ant

The War File Distribution ZIP includes Apache Ant version 1.9.4. The buildomatic Ant scripts come with Windows and Linux batch scripts pre-configured to use the bundled version of Apache Ant. You call the buildomatic Ant scripts from the command line in the following manner:

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

Linux and Mac OSX: `./js-ant <target-name>`

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

The bundled Apache Ant has an additional jar (`ant-contrib.jar`) that enables conditional logic in Ant. If you're running your own Ant, copy this jar to your Ant/lib folder.



On Linux and Solaris, the `js-ant` commands may not be compatible with all shells. If you have errors, use the bash shell explicitly. For more information, see [Bash Shell for Solaris, IBM AIX, HP UX and FreeBSD](#).

Applications Supported by the WAR File Distribution

Database and Application Server Support

The instructions in this and subsequent chapters support the following configurations:

Database	Application Server	Instructions Located In
PostgreSQL MySQL	Apache Tomcat JBossEAP/Wildfly	Current chapter
DB2	WebSphere	Installing the WAR File for WebSphere
Oracle SQL Server	WebLogic	Installing the WAR File for WebLogic

Jaspersoft recommends that you use Apache Tomcat with PostgreSQL as your repository, unless you have a strong reason to use another configuration. For version information about JVMs, application servers, databases, operating systems, and browsers, refer to the *JasperReports Server Supported Platform Datasheet*.

Operating System Support for Bash Shell

JasperReports Server is a Java Web Application, which supports all operating system platforms where Java is fully supported. However for the js-install shell scripts (described in the section below), the default shell required is the bash shell. Here is a list of shells required:

Operating System	Required Shell for js-install scripts	System Default Shell	Script to Run
Windows	CMD shell	CMD shell	js-install.bat
Linux	Bash shell	Bash shell	js-install.sh
Solaris	Bash shell	Korn shell (ksh)	js-install.sh
IBM AIX	Bash shell	Korn shell (ksh)	js-install.sh
HP UX	Bash shell	Posix shell (posix/sh)	js-install.sh
FreeBSD	Bash shell	C shell (tcsh)	js-install.sh

Installing the WAR File Using js-install Scripts

Follow this procedure to install JasperReports Server using the WAR file distribution. The js-install shell scripts, supported on Windows, Linux, and Mac, do most of the work for you.

Prerequisites for installing the WAR file

1. Install a supported version of the Java Development Kit (JDK). See the TIBCO Jaspersoft Supported Platforms Datasheet document on the [Documentation section](#) of the Jaspersoft Community website for a list.
2. Create and set the JAVA_HOME system environment variable to point to the Java JDK location.
3. Locate or install one of the following application servers. See the Jaspersoft Platform Support Guide for supported versions:
 - Apache Tomcat

- JBoss EAP or Wildfly (additional steps may be required for JBoss EAP or Wildfly. Please see [Additional Steps for Using JBoss EAP, JBoss Web Server or Wildfly](#)).
4. Locate or install the PostgreSQL, MySQL, Oracle, SQL Server, or DB2 database. If you use DB2, follow the steps in [Additional Steps for Using DB2 and js-install Scripts](#).



The target database can be on a remote server. The application server should reside on the local machine.

For an optional pre-install validation test, run `js-install.bat/sh test`. See [js-install Script Test Mode](#) for more information.

To install the WAR file using js-install scripts

The scripts are intended for the bash shell.



If installing to non-Linux Unix platforms such as IBM AIX, FreeBSD, or Solaris, the bash shell is required for using the js-install scripts.

1. Extract all files from `js-jrs_9.0.0_bin.zip`. Choose a destination, such as:
 - On Windows: `C:\Jaspersoft`
 - On Linux: `/home/<user>`
 - On Mac: `/Users/<user>`

The directory, on `Linux_bin`, appears in the file location you choose.

2. Copy the `<dbType>_master.properties` file for your database from `sample_conf` and paste it to `builddomatic`:
 - Copy from: `<js-install>/builddomatic/sample_conf/`
 - Paste to: `<js-install>/builddomatic`

For example, if your database is PostgreSQL, copy `postgresql_master.properties` to `<js-install>/builddomatic`.

3. Rename the file that you copied to `default_master.properties`.
4. Edit the `default_master.properties` file to add the settings for your database and application server. [Sample Values for the default_master.properties File](#) lists sample property values for each supported database.

Sample Values for the default_master.properties File

Database	Sample Property Values
PostgreSQL	<pre> appServerType=tomcat [jboss-eap-7, wildfly, skipAppServerCheck] appServerDir=c:\\Program Files\\Apache Software Foundation\\Tomcat 9.0 dbHost=localhost dbUsername=postgres dbPassword=postgres </pre>
MySQL	<pre> appServerType=tomcat [jboss-eap-7, wildfly, skipAppServerCheck] appServerDir=c:\\Program Files\\Apache Software Foundation\\Tomcat 9.0 dbUsername=root dbPassword=password dbHost=localhost </pre>
Standard Oracle options	<pre> appServerType=tomcat [jboss-eap-7, wildfly, skipAppServerCheck] appServerDir=c:\\Program Files\\Apache Software Foundation\\Tomcat 9.0 dbUsername=jasperserver dbPassword=password sysUsername=system sysPassword=password dbHost=hostname </pre>
Additional options for Oracle CDB with common users	<p data-bbox="500 1402 1401 1476">If you are using Oracle CDB and you want to use a common Jaspersoft user, then use the settings for Oracle with the following changes:</p> <pre data-bbox="529 1518 922 1549">dbUsername=c##jasperserver</pre> <pre data-bbox="529 1623 699 1654">sid=orclcdb</pre>
	<p data-bbox="500 1717 938 1749">If you are using sample databases:</p>

Database	Sample Property Values
	<pre data-bbox="529 310 1003 342">foodmart.dbUsername=c##foodmart</pre> <pre data-bbox="529 415 1003 447">sugarcrm.dbUsername=c##sugarcrm</pre>
DB2	<pre data-bbox="529 548 1247 779">appServerType=tomcat [jboss-eap-7, wildfly, skipAppServerCheck] appServerDir=c:\\Program Files\\Apache Software Foundation\\Tomcat 9.0 dbUsername=db2inst1 dbPassword=password dbHost=localhost</pre> <p data-bbox="496 831 1386 905">If you use DB2, follow the steps in Additional Steps for Using DB2 and js-install Scripts</p>
SQL Server	<pre data-bbox="529 974 1247 1163">appServerType=tomcat [jboss-eap-7, wildfly, skipAppServerCheck] appServerDir=c:\\Program Files\\Apache Software Foundation\\Tomcat 9.0 dbUsername=sa dbPassword=sa</pre> <pre data-bbox="529 1247 776 1278">dbHost=localhost</pre>
Additional options for all databases	<pre data-bbox="496 1358 987 1390">(Optional) jrws.deploy.webapps=true</pre> <pre data-bbox="496 1421 1224 1453">(Optional) jrws.repo.url=http://localhost:8080/repo</pre> <pre data-bbox="496 1484 1224 1516">(Optional) jrws.jrio.url=http://localhost:8080/jrio</pre> <p data-bbox="496 1547 1414 1652">By default, JasperReports Web Studio applications are installed into the same application server as JasperReports Server, as configured in the <code>appServerDir</code> property.</p> <p data-bbox="496 1684 1256 1751">If you do not want to deploy the JasperReports Web Studio applications, set</p>

Database	Sample Property Values
	<pre data-bbox="496 296 873 327">jrws.deploy.webapps=false.</pre> <p data-bbox="496 354 1370 386">In that case the integrated JasperReports Web Studio stops working.</p> <p data-bbox="496 415 1333 489">From JasperReports Server, you can still view the Open in Editor option. This option must be disabled from the context menu.</p> <p data-bbox="496 518 1354 592">For steps on how to disable the Open in Editor option, refer to the <i>JasperReports Server Administrator Guide</i>.</p> <p data-bbox="496 619 1357 735">If you have already deployed JasperReports Web Studio on another server you can skip deploying it with JasperReports Server and just point to that remote applications by setting:</p> <pre data-bbox="496 758 1414 898">jrws.deploy.webapps=false jrws.repo.url=http://<jasper-reports-webstudio-host>:8080/repo jrws.jrio.url=http://<jasper-reports-webstudio-host>:8080/jrio</pre>

For the Split installation, configure the additional settings in the `default_master.properties` file as described in [Additional Buildomatic Configuration for Split Installation](#).



Note the following:

When the property `appServerType` is set to `skipAppServerCheck`, buildomatic skips any application server validation.

Backslashes in paths must be doubled in properties files, for example:

```
appServerDir=C:\\Apache Software Foundation\\Tomcat 9.0.
```

The `dbUsername` must be the same as the Oracle user name. In addition, buildomatic does with the “`sys as sysdba`” syntax.

For Oracle without CDB with common users, do not use the `c##jasperserver` `dbUsername`. Use the standard `jasperserver` `dbUsername` instead.



On Linux, if Tomcat is installed using `apt-get`, `yum`, or `rpm`, see [Tomcat Installed Using apt-get/yum](#).

5. In case DB2, SQL Server or Oracle databases are used you must install the required JDBC driver. For steps to install, see [Installing Database Vendor JDBC Drivers](#).
6. Run the `js-install` script:
 - a. Start your database server.
 - b. Stop your application server.
 - c. Open Command Prompt as Administrator on Windows or open a terminal window on Linux and Mac OSX.
 - d. Run the `js-install` script for the version and files that you want, as shown in the following table:

Commands	Description
<code>cd <js-install>/buildomatic</code>	
<code>js-install.bat</code> (Windows) <code>./js-install.sh</code> (Linux and Mac OSX)	Installs JasperReports Server and JasperReports Web Studio, sample data, and sample databases (foodmart and sugarcrm)
<code>js-install.bat minimal</code> (Windows) <code>./js-install.sh minimal</code> (Linux and Mac OSX)	Installs JasperReports Server and JasperReports Web Studio, but not the sample data and sample databases

If you encounter errors during the `js-install` script execution, see [Error Running js-install Scripts \(js-install.bat/sh\)](#).

7. Set Java JVM Options (required), as described in [Setting JVM Options for Application Servers](#).
8. Set up the license (required) as described in [Setting Up the JasperReports Server License](#).



To view the output log, look in: `<js-install>/buildomatic/logs/js-install-
<date>.log`



Installing JasperReports Server automatically generates encryption keys that reside on the file system. These keys are stored in a dedicated Jaspersoft keystore. Make sure that this keystore is properly secured and backed up, as described in the *JasperReports Server Security Guide*.

Installing Chrome/Chromium

You need to install and configure Chrome/Chromium to export the reports and dashboards to PDF and other output formats.

For information about configuring Chrome/Chromium in JasperReports Server, see the *JasperReports Server Administrator Guide*.

Additional Steps for Using DB2 and js-install Scripts

The buildomatic scripts cannot automatically connect to a remote DB2 database and carry out Admin operations, so you have to perform additional steps to create the databases.

The DB2 client software, db2 or db2cmd, can be used to interact with DB2.

1. Enter commands similar to the ones below in the DB2 command window to create and initialize the repository database, called jsprsrvr in DB2 to conform to the 8-character limitation:

```
db2 create database jsprsrvr using codeset utf-8 territory us pagesize 16384
```

2. Enter the following commands to create and initialize the JSAudit database jsaudit in DB2 to conform to the 8-character limitation:

```
db2 create database jsaudit using codeset utf-8 territory us pagesize 16384
```

3. (Optional) Run the following commands in the DB2 command window if you want to install sample databases:

```
db2 create database sugarcrm
db2 create database foodmart
```

4. Continue installing JasperReports Server as described in [Installing the WAR File Using js-install Scripts](#).

Further considerations:

- If JasperReports Server is deployed on the same host as DB2, delete the following file to avoid conflicts:
<db2>/SQLLIB/java/db2jcc.jar

Additional Steps for Using JBoss EAP, JBoss Web Server or Wildfly

If you're using JBoss EAP, JBoss Web Server (JWS), or Wildfly as your application server and Oracle, SQL Server, or DB2 as your database, an additional set of steps is required to handle the JDBC driver. If you're using a driver different from the one supplied with JasperReports Server, you should have already downloaded a JDBC driver jar for your database type. (See [Working With JDBC Drivers](#), if you have not yet done this.)



There is a JDBC JAR issue that causes an installation error with JBoss EAP 7.0.0 with JasperReports Server 7.9 and 7.8. Because the solution is to remove the JAR, this prevents using Elasticsearch with this combination of servers. This issue does not affect JBoss EAP 7.1.0 or 7.2.0. For more information, see [JBoss 7.2.0 Startup JDBC Version Error](#).

For other issues and work-arounds for JBoss EAP, see [JBoss Modifications](#) in the Troubleshooting appendix.

JBoss EAP/Wildfly Installation

You need to make an explicit reference to your JDBC driver file name so that JBoss EAP/Wildfly knows the exact file name. For example, if you are using Oracle as your database, you need to do the following:

1. Update your `default_master.properties` file to specify the exact name (`artifactId` and `version`) of your JDBC driver:

2. Edit:

```
<js-install>/buildomatic/default_master.properties
```

- a. Look for the section "Setup JDBC Driver".
- b. Uncomment and edit these two lines:

```
# maven.jdbc.artifactId=ojdbc6  
# maven.jdbc.version=11.2.0.3
```

So that they look like this:

```
maven.jdbc.artifactId=ojdbc6  
maven.jdbc.version=11.2.0.3
```

(This works for a driver with the filename: `ojdbc6-11.2.0.3.jar`)

3. Edit your `jboss-deployment-structure.xml` file so that the JDBC filename is specified:

- a. Edit:

```
<js-install>/buildomatic/install_resources/jboss7/jboss-deployment-  
structure.xml
```

- b. Look for the section "Setup JDBC Driver".

- c. Uncomment and edit the line for your database type, as shown in the following example:

```
<!-- <resource-root path="WEB-INF/lib/ojdbc6-11.2.0.3.jar" use-  
physical-code-source="true"/> -->
```

So that it looks like this:

```
<resource-root path="WEB-INF/lib/ojdbc6-11.2.0.3.jar" use-physical-  
code-source="true"/>
```

(This works for a driver with the filename: `ojdbc6-11.2.0.3.jar`)



If your JDBC driver filename does not have a version number, you may need to rename the file and give it a version number.

For instance, if you are using SQL Server and you have a file named:

```
sqljdbc6.jar
```

Then, you can rename it with a "dummy" version number:

```
sqljdbc6-1.6.jar
```

Then the `artifactId` and `version` can look like this:

```
maven.jdbc.artifactId=sqljdbc6
```

```
maven.jdbc.version=1.6
```

JBoss Web Server Installation

Installation for JBoss Web Server (JWS) is similar to installing Tomcat.

In the `default_master.properties` file:

- The `appServerType` can be left as **tomcat**.
- The `appServerDir` should be set to the `../tomcat` directory.

For example:

```
appServerDir = /opt/jws-5.7/tomcat
```



All the databases are verified to work with JasperReports Server on JWS 5.7.2. Any DB2, MySQL, Oracle, PostgreSQL, SQL Server versions, listed in the *JasperReports Server Platform Support Guide* should work.

Additional Steps for Using the IBM JDK

If you are using the IBM JDK, you need to set OWASP to use the correct Pseudo-random Number Generator (PRNG). To do this before installation, you can modify the WAR file as follows:

1. The WAR file is an archive format in a single file.
 - a. Extract the `Websphere.jrs.csrfguard.properties` file using the following command:

```
cd <js-install>
"%JAVA_HOME%/bin/jar" xf jasperserver-pro.war WEB-
INF/csrf/Websphere.jrs.csrfguard.properties
```

This creates the `WEB-INF/csrf` folder in the current location and places the extracted file there.

- b. Rename the file from `Websphere.jrs.csrfguard.properties` to `jrs.csrfguard.properties` using the following command:

```
mv ./WEB-INF/csrf/Websphere.jrs.csrfguard.properties ./WEB-
INF/csrf/jrs.csrfguard.properties
```

2. After you have modified the file, replace it in the WAR file archive using the following commands.

```
cd <js-install>
```

```
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-
INF/csrf/jrs.csrfguard.properties
```

Starting the Server

To run JasperReports Server

Start your application server using one of these commands:

Tomcat:	Windows	<code><tomcat>\bin\startup.bat</code>
---------	---------	---

	Linux and Mac OSX	<tomcat>/bin/startup.sh
JBoss:	Windows	<jboss>\bin\standalone.bat
	Linux and Mac OSX	<jboss>/bin/standalone.sh

To view the JasperReports Server application logs, see [Log Files](#).

Logging into the Server

After JasperReports Server starts up, log in by going to this URL:

`http://<hostname>:8080/jasperserver-pro`

Example:

`http://localhost:8080/jasperserver-pro`

`http://jasperserver.example.com:8080/jasperserver-pro`

The login page appears after compiling the necessary JSP files (this may take a few moments).

Use the following credentials to log into JasperReports Server:

User ID	Password	Description
superuser	superuser	System-wide administrator
jasperadmin	jasperadmin	Administrator for the default organization

If you logged in successfully, your JasperReports Server home page appears.



When you complete the evaluation or testing of your JasperReports Server instance, change the administrator and superuser passwords (jasperadmin and superuser) and remove any sample end-users. Leaving the default passwords and end-users in place weakens the security of your installation.

Refer to the JasperReports Server User Guide to begin adding reports and other objects to the server.

JasperReports Server Heartbeat

After your initial login, you're asked to opt in to the JasperReports Server Heartbeat. The heartbeat helps Jaspersoft understand customer installation environments to improve our products. If you choose to enable the heartbeat, an HTTPS call at server startup time sends information like this to Jaspersoft:

- Operating System and JVM type and version
- Application Server and Database type and version
- JasperReports Server type and version
- Unique, anonymous identifier value

You can manually enable or disable the heartbeat by modifying the following property file `jasperserver-pro/WEB-INF/js.config.properties`. To disable the heartbeat, set the `heartbeat.enabled` property to `false`:

```
heartbeat.enabled=false
```

For additional information about enabling and disabling the heartbeat component, see the [JasperReports Server Administrator Guide](#).

Log Files

Runtime log files contain important information about JasperReports Server operations. Depending on your application server, the log output goes to one of the following files.

Tomcat:	<code><tomcat>/webapps/jasperserver-pro/WEB-INF/logs/jasperserver.log</code>
---------	--

JBoss:	<code><jboss>/standalone/deployments/jasperserver-pro.war/WEB-INF/logs/jasperserver.log</code>
--------	--

To view the log file, you must have access to the file system where JasperReports Server is installed. This section describes the settings that control the information JasperReports Server writes to its logs.

Managing Log Settings

To set the current logging levels

1. Log in as system administrator (superuser by default).
2. Select Manage > Server Settings and choose Log Settings in the left-panel.
3. In the Log Settings panel, use the drop-down selectors to change the log level for each class being logged.

For more information about system logging, see the JasperReports Server Administrator Guide.

Troubleshooting Your Server Configuration

This section helps you troubleshoot the most common installation problems.

Startup Problems

If you encounter a problem while trying to run a new JasperReports Server, it may be due to an incorrect database configuration. Another reason could be an error in the application server configuration files. For information about resolving these types of errors, see [Troubleshooting](#).

Error Running a Report

If you have trouble running reports in your new JasperReports Server instance, see [Error Running a Report](#) in [Troubleshooting](#).

Error Running js-install Scripts (js-install.bat/sh)

The js-install script creates an output log that captures standard output and error output. If you encounter problems during the execution of the script, or if you want to remember which options you chose, open the output log file.

To troubleshoot problems running js-install scripts

1. Open the output log file available at:
`<js-install>/buildomatic/logs/js-install-<date>-<number>.log`
2. Try to find the first error encountered by the `js-install` steps.
 - Go to the end of the output log.
 - Scroll back through lines of error messages until you find the first error logged. Typically, this error causes more errors later in the log.
 - Finding the original error is the way to understand the problem. However, this can often be tricky because Java stack traces with the Spring application component framework can make the error output quite long.
3. Incorrect settings in the `default_master.properties` file cause most problems, which you can correct by editing your `default_master.properties` settings. Common errors are:
 - Typos in the path for the application server
 - Misspelling the hostname or password for the database

To recreate your default_master.properties settings

1. Open the file `<js-install>/buildomatic/default_master.properties`, make corrections, and save it.
2. Re-run the `js-install` script.
The `js-install` script uses the current values in the `default_master.properties` file.

To help isolate errors, run the `js-install` scripts in `test` mode.

js-install Script Test Mode

You can run the `js-install` and `js-upgrade` scripts in `test` mode using the `test` option. In `test` mode, the `js-install` scripts check your `default_master.properties` settings and validate the application server location and connection to the specified database. Using `test` mode can help debug issues, such as an incorrect database password. Your system isn't altered when executing the script in `test` mode.

To run the js-install script in test mode on Windows

1. Navigate to the buildomatic directory:
`cd <js-install>/buildomatic`
2. Enter the following command to run the js-install script in test mode:
`js-install.bat test`

To run the js-install script in test mode on Linux or Mac OSX

1. Navigate to the buildomatic directory:
`cd <js-install>/buildomatic`
2. Enter the following command to run the js-install script in test mode:
`./js-install.sh test`

Problem Connecting to a Cloud Database Instance

A cloud database instance (such as Amazon EC2) typically disables unused IP ports. When the `js-install` script runs, it validates the database hostname using the built-in `ant` operation `<isreachable>`. This operation is similar to a network ping and may stop responding if the port is unavailable. In this case, the `validateHost` step can be commented out in the `buildomatic/validation.xml` file. See the comment in the `do-pre-install-test` target.

Error Exporting Reports, Ad Hoc Views, and Dashboards

The export of Reports, Ad Hoc Views, and Dashboards fails when Tomcat is run as root in the JasperReports Server installation on Linux. The Tomcat log file displays an error, for example:

```
2020-06-11T17:32:19,031 ERROR SecureExceptionHandlerImpl,http-nio-8080-  
exec-8:116 -  
com.github.kklisura.cdt.launch.exceptions.ChromeProcessTimeoutException:
```

```
Failed while waiting for chrome to start: Timeout expired! Chrome output:  
[0611/173218.897370:ERROR:zygote_host_impl_linux.cc(89)] Running as root  
without --no-sandbox is not supported. See https://crbug.com/638180.
```

To resolve this, set the following property in the `jasperreports.properties` file:

- `net.sf.jasperreports.chrome.argument.no-sandbox=true`

After setting this property, restart JasperReports Server to enable it.

For information about configuring this property, see the *JasperReports Server Administrator Guide*.

Installing the WAR File Manually

You may need to install the WAR file manually when you cannot use the `js-install` scripts.

The manual buildomatic steps described in this procedure execute the same Ant targets as the `js-install` script (`js-install.sh/.bat`). The procedure shows which buildomatic targets to run manually if you are unable to use the `js-install` scripts.

To install the WAR file distribution using manual buildomatic steps

1. Start your database server.
2. Stop your application server.
3. Create and edit a `default_master.properties` file to add the settings in for your database and application server as described in [Installing the WAR File Using js-install Scripts](#).
4. Open a Command Prompt as Administrator on Windows or open a terminal window on Linux or Mac. Run the following commands.

Buildomatic Targets to Execute to Install the WAR File

Commands	Description
<code>cd <js-install>/buildomatic</code>	Makes the buildomatic directory your current directory.
<code>js-ant create-js-db</code>	Creates the JasperReports Server repository database.
<code>js-ant create-sugarcrm-db</code> <code>js-ant create-foodmart-db</code>	(Optional) Creates the sample databases.
<code>js-ant load-sugarcrm-db</code> <code>js-ant load-foodmart-db</code>	(Optional) Loads sample data into the sample databases.
<code>js-ant update-foodmart-db</code>	(Optional) Initializes the sample databases
<code>js-ant init-js-db-pro</code> <code>js-ant import-minimal-pro</code>	Initializes the jasperserver database, loads core application data. Running <code>js-ant import-minimal-pro</code> is mandatory. The server needs this data to function.
<code>js-ant import-sample-data-pro</code>	(Optional) Loads the demos that use the sample data.
<code>js-ant deploy-webapp-pro</code>	Configures and deploys the WAR file to Tomcat or JBoss.
<code>js-ant deploy-jrws</code>	Deploys only JasperReports Web Studio apps into the application server configured using the <code>appServerDir</code> , <code>jrws.repo.url</code> and <code>jrws.jrio.url</code> properties in the <code>default_master.properties</code> file.
<code>js-ant create-audit-db</code>	(Optional) Creates the audit database. Required only for the Split installation.
<code>js-ant init-audit-db-pro</code>	(Optional) Initializes the audit database. Required only for the Split installation.



On non-Linux Unix platforms, the js-ant commands may not be compatible with all shells. If you have errors, use the bash shell explicitly. For more information, see [Bash Shell for Solaris, IBM AIX, HP UX and FreeBSD](#).

If you encounter an error when running `create-sugarcrm-db`, `create-foodmart-db`, or `create-js-db`, you can create the JasperReports Server database manually using the database administration tool for your particular database type. To create the JasperReports Server database manually for PostgreSQL, MySQL, Oracle, SQL Server, or DB2, refer to [Manually Creating the JasperReports Server Database](#).

If you have previously installed the databases, you can drop the old versions and then recreate the databases. To do this, run the following drop commands before running the commands in [Buildomatic Targets to Execute to Delete Sample Databases](#)

Buildomatic Targets to Execute to Delete Sample Databases

Commands	Description
<code>js-ant drop-sugarcrm-db</code> <code>js-ant drop-foodmart-db</code>	(Optional) Deletes the sample databases.
<code>js-ant drop-js-db</code>	(WARNING) Deletes the JasperReports Server repository database. Only run this command if you intend to recreate the <code>jasperserver</code> database
<code>js-ant drop-audit-db</code>	(Optional) Deletes the audit database. Required only for the Split installation, or if the previous installation was Split, or for the database clean up after a Split installation.

5. Set Java JVM Options (required) as described in [Setting JVM Options for Application Servers](#).
6. Set up the license (required) as described in [Setting Up the JasperReports Server License](#).



Installing JasperReports Server automatically generates encryption keys that reside on the file system. These keys are stored in a dedicated Jaspersoft keystore. Make sure that this keystore is properly secured and backed up, as described in the *JasperReports Server Security Guide*.

JVM Options, License Setup, Working with JDBC Drivers

This chapter contains the following sections:

- [Setting JVM Options for Application Servers](#)
- [Setting Up the JasperReports Server License](#)
- [Working With JDBC Drivers](#)
- [Locating and Changing Buildomatic Configuration Files](#)
- [Configuring Report Scheduling](#)
- [Updating XML/A Connection Definitions](#)

Setting JVM Options for Application Servers



The settings in this section apply specifically to the Oracle/Sun JVM. Other JVMs may or may not have equivalent settings. For a list of supported JDK/JVMs and application servers, see the TIBCO JasperSoft Platform Support document.

You may need to set the following options for your JVM:

- Memory – Java Virtual Machine (JVM) runtime parameters normally need to be explicitly set so that the memory settings have values larger than the default settings. The options and values depend on your version of Java and the application server that you use. You may need to increase the memory assigned for the JVM according to your usage.



If JasperReports Web Studio is deployed in the same application server as JasperReports Server, the memory demand increases. Hence, the memory assigned for Tomcat must be adjusted. It is recommended to increase Xmx at least by 0.5 GB.

- Garbage collection – You may need to tune garbage collection for your JVM, depending on your memory and CPU usage as well as JasperReports Server throughput. Different collectors have different performance characteristics. Consult the documentation for your JVM for information on available collectors.
- Date format in Java 11 – Java 11 changed the default date format to conform to CLDR 33. If you are running on Java 11 and want backwards compatibility with Java 8 date formatting, you need to set
-Djava.locale.providers=COMPAT on all nodes.
- UTF-8 support for Oracle – If you need to support UTF-8 for your Oracle database, set `defaultNChar` to `true` to ensure that the database implicitly converts all CHAR data to NCHAR when you access CHAR columns. If you do not need to support UTF-8 for your Oracle database, you can omit this setting.



For the Oracle database, setting the Oracle localization option `defaultNChar` can substantially impact the performance of JDBC queries. If you do not need to support UTF-8 for your Oracle database, you can omit this setting.

Tomcat and JBoss JVM Options

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

JVM Options on Windows (64 bit)

Options for all app servers	set JAVA_OPTS=%JAVA_OPTS% -Xms2048m -Xmx4096m -Xss2m set JAVA_OPTS=%JAVA_OPTS% -XX:+UseG1GC
-----------------------------	--

Java 11	set JAVA_OPTS=%JAVA_OPTS% -Djava.locale.providers=COMPAT
---------	--

Java 17	set JAVA_OPTS=%JAVA_OPTS% --add-opens java.base/java.io=ALL-UNNAMED --add-opens java.base/java.lang.ref=ALL-UNNAMED --add-opens java.base/java.lang=ALL-UNNAMED --add-opens java.base/java.nio.channels.spi=ALL-UNNAMED --add-opens
---------	---

JVM Options on Windows (64 bit)

```
java.base/java.nio.channels=ALL-UNNAMED --add-opens
java.base/java.nio=ALL-UNNAMED --add-opens java.base/java.security=ALL-
UNNAMED --add-opens java.base/java.text=ALL-UNNAMED --add-opens
java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens
java.base/java.util.concurrent.locks=ALL-UNNAMED --add-opens
java.base/java.util.concurrent=ALL-UNNAMED --add-opens
java.base/java.util.regex=ALL-UNNAMED --add-opens java.base/java.util=ALL-
UNNAMED --add-opens java.base/javax.security.auth.login=ALL-UNNAMED --
add-opens java.base/javax.security.auth=ALL-UNNAMED --add-opens
java.base/jdk.internal.access.foreign=ALL-UNNAMED --add-opens
java.base/sun.net.util=ALL-UNNAMED --add-opens java.base/sun.nio.ch=ALL-
UNNAMED --add-opens java.rmi/sun.rmi.transport=ALL-UNNAMED --add-opens
java.base/sun.util.calendar=ALL-UNNAMED
```

For Oracle (optional)	set JAVA_OPTS=%JAVA_OPTS% -Doracle.jdbc.defaultNChar=true
--------------------------	---

Note: The java opts from the first line "Options for all app servers" should be added, and then an additional line is added either for Java 11 or Java 17.

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

JVM Options on Linux and Mac OSX (64 bit)

Options for all app servers	export JAVA_OPTS="\$JAVA_OPTS -Xms2048m -Xmx4096m -Xss2m" export JAVA_OPTS="\$JAVA_OPTS -XX:+UseG1GC"
-----------------------------------	--

Java 11	export JAVA_OPTS="\$JAVA_OPTS -Djava.locale.providers=COMPAT"
---------	---

Java 17	export JAVA_OPTS="\$JAVA_OPTS --add-opens java.base/java.io=ALL-UNNAMED --add-opens java.base/java.lang.ref=ALL-UNNAMED --add-opens java.base/java.lang=ALL-UNNAMED --add-opens java.base/java.nio.channels.spi=ALL-UNNAMED --add-opens java.base/java.nio.channels=ALL-UNNAMED --add-opens java.base/java.nio=ALL-
---------	---

JVM Options on Linux and Mac OSX (64 bit)

```
UNNAMED --add-opens java.base/java.security=ALL-UNNAMED --add-opens
java.base/java.text=ALL-UNNAMED --add-opens
java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens
java.base/java.util.concurrent.locks=ALL-UNNAMED --add-opens
java.base/java.util.concurrent=ALL-UNNAMED --add-opens
java.base/java.util.regex=ALL-UNNAMED --add-opens java.base/java.util=ALL-
UNNAMED --add-opens java.base/javax.security.auth.login=ALL-UNNAMED --
add-opens java.base/javax.security.auth=ALL-UNNAMED --add-opens
java.base/jdk.internal.access.foreign=ALL-UNNAMED --add-opens
java.base/sun.net.util=ALL-UNNAMED --add-opens java.base/sun.nio.ch=ALL-
UNNAMED --add-opens java.rmi/sun.rmi.transport=ALL-UNNAMED --add-opens
java.base/sun.util.calendar=ALL-UNNAMED
```

For Oracle (optional)	<code>export JAVA_OPTS="\$JAVA_OPTS -Doracle.jdbc.defaultNChar=true"</code>
--------------------------	---



The java opts from the first line "Options for all app servers" should be added, and then an additional line is added either for Java 11 or Java 17.

You can set JVM options multiple ways. Sections [Changing JVM Options for Tomcat as a Windows Service](#) and [Setting JVM Options for Application Servers](#) present step-by-step instructions for performing this task. Alternatively, you can add your JAVA_OPTS settings to any of the following files.

File	Add JVM Options After This Line on Windows
<tomcat>\bin\setclasspath.bat	<pre>set JAVA_ENDORSED_ DIRS=%BASEDIR%\common\endorsed</pre>
<tomcat>\bin\setenv.bat	JAVA_OPTS setting can go anywhere in this file.
<jboss>\bin\standalone.conf.bat	<p>Find the existing JAVA_OPTS line, remove the default memory settings from this line, and add a new line with the recommended JAVA_OPTS after this line.</p> <p>For example, you might remove the following default settings:</p> <pre>-Xms64m -Xmx512m -XX:MetaspaceSize=96M - XX:MaxMetaspaceSize=256m</pre> <p>Then add the recommended settings on a new line. We recommend that you do not set MaxMetaspaceSize.</p>

File	Add JVM Options After This Line on Linux
<tomcat>/bin/setclasspath.sh	<pre>JAVA_ENDORSED_ DIRS="\$BASEDIR"/common/endorsed</pre>
<tomcat>/bin/setenv.sh	JAVA_OPTS setting can go anywhere in this file.
<jboss>/bin/standalone.conf	<p>Find the existing JAVA_OPTS line, remove the default memory settings from this line, and add a new line with the recommended JAVA_OPTS after this line.</p> <p>For example, you might remove the following default settings:</p> <pre>-Xms64m -Xmx512m -XX:MetaspaceSize=96M - XX:MaxMetaspaceSize=256m</pre> <p>Then add the recommended settings on a new line. We recommend that you do not set MaxMetaspaceSize.</p>

Changing JVM Options for Tomcat as a Windows Service

If you installed JasperReports Server to use Tomcat running as a Windows service, you can set Java options on the Java Tab of the Tomcat Properties dialog:

1. Launch the Tomcat configuration application. If you installed the bundled Tomcat, you can do this by going to the <js-install>/apache-tomcat/bin directory and double-clicking the jasperreportsTomcat.exe file. (If you have multiple instances of JasperReports Server installed, the file name will be of the form jasperreportsTomcatnum<number>.exe, for example, jasperreportsTomcatnum2.exe.) If you installed Tomcat using an existing Windows service, look for an .exe file in the same location, with the same name as your Tomcat service, or select the service from the Windows Start menu:

Start > Programs > Apache Tomcat > Configure Tomcat (Run as administrator)

2. In the Apache Tomcat Properties dialog, click the Java tab.
3. In the Java Options field, add your JAVA_OPTS values according to the tables above. Enter only the options preceded by -X or -D, not set JAVA_OPTS=%JAVA_OPTS%. Enter only one Java option setting per line.
4. For instance, add options as follows:

```
-Xms2048m  
-Xmx4096m  
-Xss2m
```

5. Click Apply, then click OK.
6. Stop and restart Tomcat.

Changing JVM Options for Bundled Tomcat on Linux

If you installed the bundled Tomcat, you can set Java options by editing the appropriate Tomcat configuration script. The steps to change JVM options are:

1. Open the following file for editing:

```
cd <js-install>/apache-tomcat/scripts/ctl.sh
```

2. Look for the `start_tomcat()` function and locate the `JAVA_OPTS` variable inside it.
3. Modify the `JAVA_OPTS` values according to the tables above. For example:

```
start_tomcat() {
    is_tomcat_running
    ...
    export JAVA_OPTS="-Xms2048m -Xmx4096m"
    export JAVA_OPTS="-Xss2m -XX:+UseG1GC"
    ...
}
```



There may be more than one occurrence of the `Java_OPTS` variable in the `ctl.sh` file. Make sure you edit the instance inside the `start_tomcat()` function.

4. Save and close the `ctl.sh` file.
5. Stop and restart PostgreSQL and Tomcat as described in [Starting and Stopping the Server](#).

Setting Up the JasperReports Server License

JasperReports Server requires a license and comes with an evaluation license. Please contact [Jaspersoft Technical Support](https://www.jaspersoft.com/support) (<https://www.jaspersoft.com/support>) or your sales representative to get your commercial license.

The license file is in the following location:

```
<js-install>/jasperserver.license
```

The license file specifies the terms of your license, such as the following:

- Expiration date, number of users, and/or number of CPUs

- Features licensed separately from the basic commercial license, such as multi-tenancy

Jaspersoft receives information about your system periodically. The information is used only to monitor compliance with your license. No personal information is collected or transmitted.

Default License Configuration for All Application Servers

At startup JasperReports Server automatically looks for the `jasperserver.license` file in the home directory of the system user running the application server. [License Locations](#) lists the application server user home directories for supported operating systems.

To configure the license:

1. Stop the application server.
2. Copy the `jasperserver.license` file in `<js-install>` to the directory for your operating system.

License Locations

Operating System	
Linux	<code>/home/<user>/</code>
Mac OSX	<code>/Users/<user>/</code>
Windows 10 installed from WAR file	<code>C:\Users\<user>\</code>
Windows 10 installed from the binary installer	<code>C:\Users\</code>
Windows 10 using an existing Tomcat Windows service	<code>C:\WINDOWS\system32\config\systemprofile</code>

User-Defined License Location

If you prefer to put your license in another directory, modify your application server startup script to set a `JAVA_OPT` value to explicitly point to that directory.

Alternate License Setup for Tomcat

If your license is not located in the home directory of the application server user, you can set a `JAVA_OPT` value to explicitly point to your license.

On Windows:

1. In the file `<tomcat>\bin\setclasspath.bat`, locate the following line:

```
set JAVA_ENDORSED_DIRS=%BASEDIR%\common\endorsed
```

Alternatively, create an empty file called `<tomcat>/bin/setenv.bat`.

2. Below that line or in the new file, insert the following line:

```
set JAVA_OPTS=%JAVA_OPTS% -Djs.license.directory="<js-install>"
```

For example:

```
set JAVA_OPTS=%JAVA_OPTS% -Djs.license.directory="C:\jasperserver-  
bin"
```

On Linux and Mac OSX:

1. In the file `<tomcat>/bin/setclasspath.sh`, locate the following line:

```
JAVA_ENDORSED_DIRS="$BASEDIR"/common/endorsed
```

Alternatively, create an empty file called `<tomcat>/bin/setenv.sh`.

2. Below that line or in the new file, insert the following line:

```
export JAVA_OPTS="$JAVA_OPTS -Djs.license.directory=<js-install>"
```

For example:

```
export JAVA_OPTS="$JAVA_OPTS -  
Djs.license.directory=/home/user/jasperserver-bin"
```


Alternate License Setup for Bundled Tomcat as a Windows Service

The Windows binary installer installs the bundled Tomcat component as a Windows Service by default.

To specify a specific folder to hold the `jasperserver.license`:

1. Open the following file for editing:
`cd <js-install>/apache-tomcat/bin/service.bat`
2. Look for the second line of two lines that set JVM options, specifically the line which contains the license string. For example, -
`Djs.license.directory=C:\Jaspersoft\jasperreports-server-9.0.0.`
3. Update the line to point to your license location, for example:
`-Djs.license.directory=C:\MyLicenses`

Because Tomcat is installed as a service, you need to re-install the service. From a Windows Command shell, enter these commands (Note: the cmd shell will disappear when these commands are run. You need to open a new cmd shell for each command.). To open a cmd shell: Start Menu > Run... > cmd:

```
cd <js-install>\apache-tomcat\scripts
serviceinstall.bat REMOVE
serviceinstall.bat INSTALL
```

The Tomcat service is removed and then installed. After execution of these commands, the service is running.

Alternate License Location for Existing Tomcat as a Windows Service

Windows 10

If you have an existing Tomcat as a Windows Service under Windows 10, copy your license to the home folder for the SYSTEM user. The location is:

```
C:\WINDOWS\system32\config\systemprofile\jasperserver.license
```

Alternate License Setup for JBoss

If your license is not located in the home directory of the application server user, you can set a `JAVA_OPT` value to explicitly point to your license.

On Windows

1. In the file `<jboss>\bin\standalone.conf.bat`, locate the following line:

```
set JAVA_OPTS=%JAVA_OPTS% -Dprogram.name=%PROGNAME%
```

2. Below that line, insert the following line:

```
set JAVA_OPTS=%JAVA_OPTS% -Djs.license.directory="<js-install>"
```

For example:

```
set JAVA_OPTS=%JAVA_OPTS% -Djs.license.directory="C:\jasperserver-  
bin"
```

On Linux and Mac OSX

1. In the file `<jboss>/bin/standalone.conf`, locate the following line:

```
export JAVA_OPTS="$JAVA_OPTS -Dprogram.name=$PROGNAME"
```

2. Below that line, insert this line:

```
export JAVA_OPTS="$JAVA_OPTS -Djs.license.directory=<js-install>"
```

For example:

```
export JAVA_OPTS="$JAVA_OPTS -  
Djs.license.directory=/home/user/jasperserver-bin"
```

Working With JDBC Drivers

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

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.

PostgreSQL Example

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

```
postgresql-42.2.5.jar
```

For instance, to change the default driver used by PostgreSQL from type `jdbc4.2` to `jdbc4.1`:

Procedure

1. Download `postgresql-42.2.5.jre7` from <https://jdbc.postgresql.org/download.html> and save it under `buildomatic/conf_source/db/postgresql/jdbc`.
2. Edit your `default_master.properties` file, <js-install>Edit your `default_master.properties` file, <js-install>/`buildomatic/default_master.properties` as follows:
follows:

Uncomment and change:

```
# maven.jdbc.version=42.2.5
```

To:

```
maven.jdbc.version42.2.5.jre7
```

When you next run a buildomatic command, such as `deploy-webapp-pro`, the `jdbc4.1` driver is copied to your application server.

MySQL Example

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

```
mariadb-java-client-2.5.3.jar
```

If for instance, you want to use a JDBC driver built and distributed by the MySQL project, such as `mysql-connector-java-8.0.20-bin.jar`, then you first need to:

1. Download the driver from the MySQL Connector/J download location:
2. Next, change your buildomatic configuration properties to point to this new driver.

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

- a. Edit your default_master.properties file:

`<js-install>/buildomatic/default_master.properties`

- b. 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
```

Installing Database Vendor JDBC Drivers

JasperReports Server no longer includes JDBC drivers for the following commercial databases:

- DB2
- Oracle
- SQL Server

You can download the driver supplied by the database vendor as described below. To do this, you must first obtain and install the driver you want, then copy that driver into buildomatic.

Download a Vendor JDBC Driver

To use the driver supplied by the database vendor, you have to download and install it.

Download Driver Jar from Vendor Website

You can download a commercial JDBC driver from the vendor's website. Here are some sites where you can download packages for supported databases:

JDBC Driver	Download Site
DB2	https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads
Oracle	https://www.oracle.com/database/technologies/appdev/jdbc-downloads.html
SQL Server	https://learn.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16

Once you have downloaded your driver, copy it to the correct location and configure your files as described in the sections below.

Collect Driver Jar from Existing Application

You may already have a JDBC driver in an application running on your network. If so, you can simply copy that driver jar to the JasperReports Server install location.

Oracle Example

Copy your Oracle driver to the following directory:

```
<js-install>/buildomatic/conf_source/db/oracle/jdbc/
```

SQL Server Example

Copy your SQL Server driver to the following directory:

```
<js-install>/buildomatic/conf_source/db/sqlserver/jdbc
```

DB2 Example

Copy your DB2 driver to the following directory:

```
<js-install>/buildomatic/conf_source/db/db2/jdbc
```

Working with Oracle RAC

You can use Oracle RAC with the TIBCO JDBC Oracle driver. This driver works with Oracle RAC with the following settings:

- Use Oracle for non-CDB connection settings as described in [Standard Oracle options](#).
- Use `js-install.bat/sh minimal` with Oracle. This option does not install sample databases.

To support additional functionality with Oracle RAC, such as load balancing with multiple servers, you need to configure your application server and manually set up the correct connection URL.

Tomcat Load Balancing Example

1. Change to the `<tomcat>/webapps/jasperserver-pro/META-INF` directory and open `context.xml` in a text editor.
2. Edit the `url` property by adding additional connection properties for the data sources that you want. The following example shows how to set up the connection pool to use Oracle RAC with load balancing with a primary server and three alternate servers:

```
jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on) (ADDRESS=(PROTOCOL=tcp) (HOST=myhost1) (PORT=1525)) (ADDRESS=(PROTOCOL=tcp) (HOST=myhost2) (PORT=1525)) (CONNECT_DATA=(SERVICE_NAME=myserviceDB1)))
```

JBoss EAP/WildFly Load Balancing Example

1. Change to the <jboss-install>/standalone/deployments/jasperserver-pro.war/WEB-INF directory and open js-jboss7-ds.xml in a text editor.
2. Edit the connection-url tag for the data sources that you want. The following example shows how to set up a connection pool to use PostgreSQL RAC with load balancing with a primary server and three alternate servers.

```
<datasource jta="false" jndi-name="java:/jdbc/jasperserver"
pool-name="jasperserver" enabled="true" use-ccm="false">
  <connection-url>jdbc:oracle:thin:@(DESCRIPTION=(LOAD_
BALANCE=on)(ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1525))
(ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1525))(CONNECT_DATA=
(SERVICE_NAME=myseviceDB1)))</connection-url>
  <driver>ojdbc8-23.2.0.0.jar</driver>
  <security>
    <user-name>jasperserver</user-name>
    <password>password</password>
  </security>
  <pool>
    <min-pool-size>5</min-pool-size>
    <max-pool-size>50</max-pool-size>
    <prefill>true</prefill>
  </pool>
  <validation>
    <validate-on-match>>false</validate-on-match>
    <background-validation>>false</background-validation>
    <check-valid-connection-sql>SELECT 1 FROM DUAL</check-
valid-connection-sql>
  </validation>
  <statement>
    <share-prepared-statements>>false</share-prepared-
statements>
  </statement>
</datasource>
```

For DB2, Oracle, and SQL Server, replace these with the appropriate driver details.

Application Server Copy-to Locations

When the `deploy-webapp-pro` buildomatic target is executed, it copies the JDBC driver to the following default locations:

Tomcat:	<code><tomcat>/lib</code>
JBoss:	<code><jboss>/standalone/deployments</code>
Wildfly:	<code><wildfly>/standalone/deployments</code>

Locating and Changing Buildomatic Configuration Files

The Ant-based buildomatic scripts contain support files for setting up and configuring a number of databases and application servers. This section describes the locations of some of these files and how to change their content.

Regenerating Buildomatic Settings

Whenever you change your `default_master.properties` file and re-run the `js-install` scripts (or any other buildomatic target), your generated configuration settings are automatically updated. The generated settings are in this location:

```
<js-install>/buildomatic/build_conf/default
```

The settings are regenerated automatically based on the updated timestamp on the properties file.

If you want to explicitly regenerate your configuration, run the following buildomatic targets:

```
cd <js-install>/buildomatic
js-ant clean-config
js-ant gen-config
```

The first target clears the configuration template files in the `buildomatic/build_conf/default` directory. The second re-builds the configuration settings.

Locating Buildomatic-Generated Property Files

After you set your database and application server property values, initiate buildomatic to automatically generate the database and application server configuration files needed to prepare for a JasperReports Server installation.

The generated property files are in this location:

```
<js-install>/buildomatic/build_conf/default
```

Some of the key configuration files are:

```
js.jdbc.properties
```

```
js.quartz.properties
```

```
js-jboss-ds.xml
```

```
maven_settings.xml - (used for source code build)
```

More generated property files are:

```
<js-install>/buildomatic/build_conf/default/webapp
```

Included in the /webapp directory are configuration files, such as:

```
META-INF/context.xml
```

```
WEB-INF/classes/hibernate.properties
```

```
WEB-INF/js.quartz.properties
```

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

Buildomatic Location for JasperReports Server WAR File

Buildomatic takes the JasperReports Server WAR file from the root of the `<js-install>` directory:

```
<js-install>/jasperserver-pro.war
```

When you run the `deploy-webapp-pro` target, buildomatic unpacks the war archive into your application server and copies the needed database configuration files to their appropriate locations. For instance, in the case of Tomcat:

- `<js-install/> jasperserver-pro.war`

Unpacked and copied to <tomcat>/webapps/jasperserver-pro/*

- <js-install>/buildomatic/build_conf/default/webapp/META-INF/context.xml
Copied to <tomcat>/webapps/jasperserver-pro/META-INF/context.xml
- <js-install>/buildomatic/build_conf/default/webapp/WEB-INF/classes/hibernate.properties
Copied to <tomcat>/webapps/jasperserver-pro/WEB-INF/classes/hibernate.properties
- <js-install>/buildomatic/build_conf/default/webapp/WEB-INF/js.quartz.properties
Copied to <tomcat>/webapps/jasperserver-pro/WEB-INF/js.quartz.properties
- <js-install>/buildomatic/build_conf/db/postgres/jdbc/postgresql-42.2.5.jar
Copied to <tomcat>/lib

Buildomatic Location for SQL Scripts

Buildomatic comes with SQL scripts and other utilities that support a number of databases. These files are located here:

<js-install>/buildomatic/install_resources/sql/

For example, some key files are (same pattern for additional databases):

<js-install>/buildomatic/install_resources/sql/postgresql/js-pro-create.ddl

<js-install>/buildomatic/install_resources/sql/postgresql/quartz.ddl

<js-install>/buildomatic/install_resources/sql/postgresql/upgrade-postgresql-6.0.0-6.1.0-pro.sql

<js-install>/buildomatic/install_resources/sql/postgresql/js-pro-drop.ddl

<js-install>/buildomatic/install_resources/sql/postgresql/drop-quartz.ddl



You can run these scripts manually by copying them to the location of your database client software.

Buildomatic Location for Database Creation Scripts

For most databases the buildomatic scripts can create the metadata repository database used by JasperReports Server. This is the database that stores data defining users, roles,

data sources, reports, OLAP views, domains, and other data. This database is normally named `jasperserver`.

Buildomatic attempts to create the `jasperserver` database via JDBC when the `create-js-db` target is executed. The scripts and property files used to create the `jasperserver` database are located in the following directory:

```
<js-install>/buildomatic/conf_source/db/<db_name>/scripts.properties
```

Buildomatic Location for Sample Data Catalog ZIP Files

Buildomatic includes export files that hold the JasperReports Server sample data (with examples of new features). This sample data is loaded when you run the `buildomatic` target `import-sample-data-pro`, for instance. These export files along with other important export files are located here:

```
<js-install>/buildomatic/install_resources/export/
```

Here are some key files:

```
js-catalog-<db_name>-minimal-pro.zip
```

```
js-catalog-<db_name>-pro.zip
```

Hibernate Properties Settings

After you run `buildomatic` to generate your configuration files, your `hibernate.properties` settings are in the following directory:

```
<js-install>/buildomatic/build_conf/default/webapp/WEB-INF/classes/hibernate.properties
```

Within the `jasperserver-pro` WAR file the `hibernate.properties` file is found at the following location:

```
<appserver-path>/jasperserver-pro/WEB-INF/classes/hibernate.properties
```

The `buildomatic` scripts automatically create this configuration file. When you run the `buildomatic` target `deploy-webapp-pro` this file is copied to JasperReports Server in your application server.

Hibernate property values are:

PostgreSQL:	<code>metadata.hibernate.dialect=com.jaspersoft.hibernate.dialect.PostgresqlNoBlobDialect</code>
MySQL 5.5:	<code>metadata.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect</code>
DB2:	<code>metadata.hibernate.dialect=com.jaspersoft.ji.hibernate.dialect.DB2JICustomDialect</code>
Oracle:	<code>metadata.hibernate.dialect=com.jaspersoft.ji.hibernate.dialect.OracleJICustomDialect</code>
SQL Server:	<code>metadata.hibernate.dialect=com.jaspersoft.ji.hibernate.dialect.SQLServerJICustomDialect</code>

Database Connection Configuration Files

Tomcat

When you have set up the buildomatic configuration for your database, the Tomcat context.xml is automatically created with the appropriate settings for JasperReports Server.

When you run the buildomatic target `deploy-webapp-pro`, the context.xml is automatically copied into the jasperserver-pro WAR set of files.

You can view the automatically generated context.xml at the following location:

```
<js-install>/buildomatic/build_conf/default/webapp/META-INF/context.xml
```

The final location of the context.xml is:

```
<tomcat>/webapps/jasperserver-pro/META-INF/context.xml
```

Older versions of Tomcat create a copy of the context.xml file with a changed name that is read instead of the one found in the jasperserver-pro war file. This can be confusing for Tomcat users who try to change their database settings. If you change your settings, delete the file in this location:

```
<tomcat>/conf/Catalina/localhost/jasperserver-pro.xml
```

JBoss

When you have set up the buildomatic configuration for your database, the JBoss data source definition file is automatically created with the appropriate settings for JasperReports Server.

When you run the buildomatic target `deploy-webapp-pro`, the `js-jboss-ds.xml` is automatically copied into the JBoss instance.

You can view the automatically generated `js-jboss-ds.xml` at the following location:

```
<js-install>/buildomatic/build_conf/default/js-jboss-ds.xml
```

The final location of the `js-jboss-ds.xml` is:

```
<jboss>/standalone/deployments/jasperserver-pro.war/WEB-INF/js-jboss7-ds.xml
```

When JasperReports Server is running under JBoss, a couple of INFO log messages and an XML/A connection error may occur depending on the version of JBoss you are running.

For more information, refer to the troubleshooting section [JBoss Modifications](#).

Configuring Report Scheduling

The JasperReports Server report scheduling feature is powered by the Quartz scheduler tool. Buildomatic automatically handles configuration settings for Quartz-based report scheduling.

In a deployed JasperReports Server instance, you will find the `js.quartz.properties` file in this location:

```
<app-server-path>/jasperserver-pro/WEB-INF/js.quartz.properties
```

For mail server configuration, you will find an additional property setting for authentication in this file:

```
<app-server-path>/webapps/jasperserver-pro/WEB-INF/applicationContext-  
report-scheduling.xml
```

The following configurations are discussed in this section:

- Mail Server Configuration
- Quartz Driver Delegate Class
- Report Scheduler Web URI
- Quartz Table Prefix
- Settings for import-export
- Setting Properties in the default_master.properties File
- Settings for Skipping Calendar Job Execution Immediately After Import

Mail Server Configuration Settings

You can specify email addresses to be notified when a report run is complete. To do this, configure JasperReports Server to contact an email server. The email server can be contacted via mail or by using a graph API.



By default, JasperReports Server uses mail to contact an email server. To change it to graph API, in the `js.quartz.properties` file, set `emailService.type=graph`.

Mail Server Configuration Settings Using E-mail

The following table provides the configuration information to contact an email server via mail:

Configuration File	
<code><app-server>/<deployment>/WEB-INF/js.quartz.properties</code>	
Property	Description
<code>report.scheduler.mail.sender.host</code>	The name of the computer

	hosting the mail server.
<code>report.scheduler.mail.sender.username</code>	The name of the mail server the user JasperReports Server can use.
<code>report.scheduler.mail.sender.password</code>	The password of the mail server user.
<code>report.scheduler.mail.sender.from</code>	The address for the From field on email notifications.
<code>report.scheduler.mail.sender.protocol</code>	The protocol that the mail server uses. JasperReports Server supports only SMTP. Note: Your entry must be lower case (smtp).
<code>report.scheduler.mail.sender.port</code>	The port number that the mail server uses. The default is typically 25 (other ports may not work in earlier JasperReports Server versions).



- `host`, `username`, `port` and `password` are mandatory parameters. If any of these parameters are missed, a bean is created with the default values provided in the `js.quartz.properties` file.
- If `protocol` is missed, it will set to the default value, which is SMTP.
- If the `from` address is missed, it picks the value from the `js.quartz.properties` file.

Configuration File

```
<app-server>/<deployment>/WEB-INF/applicationContext-report-scheduling.xml
```

Property	Bean	Description
javaMailProperties key="mail.smtp.auth"	reportScheduler MailSender	If your mail server requires authentication, change this property from false to true.

Mail Server Configuration Settings Using Graph API

Prerequisite for Azure Graph API login


The following are the prerequisites for Azure Graph API login:

1. Register JasperReports Server as an application on the Azure portal.
2. Obtain the details for **Resource Owner** flow and **Client Credentials** flow.
3. For **Resource Owner** flow, obtain the following details:
 - a. Username
 - b. Password
4. For **Client Credentials** flow, obtain the following details:
 - a. Client ID
 - b. Client Secret
 - c. User ID
 - d. Tenant ID
5. Define the **Mailbox Permissions and Scope**.
Ensure that the following permissions are defined in the scope:
 - a. **Mailbox**
 - b. **Mail.Send**
 - c. **Mail.ReadWrite**
6. Obtain the `resourceGroupName` configured for Azure SQL.

The following table provides the configuration information to contact an email server using a graph API:

Configuration File

<app-server>/<deployment>/WEB-INF/js.quartz.properties

Property	Description
<code>emailService.graph.oauthUrl</code>	The URL of the mail server.
<code>emailService.graph.oauthClientId</code>	The unique application or client ID assigned to JasperReports Server by Azure Active Directory (AD).
<code>emailService.graph.oauthSecret</code>	A secret string that the application uses to identify itself while requesting a token.
<code>emailService.graph.oauthTenantId</code>	An instance of Azure AD.
<code>emailService.graph.oauthUserName</code>	The username to connect to Azure.
<code>emailService.graph.oauthPassword</code>	The password to connect to Azure.
<code>emailService.graph.oauthUserId</code>	The unique object ID using which JasperReports Server is registered.
<code>clientSecretCredentialFlow</code>	<p>The authentication is done either by providing the <code>clientSecret</code> or by providing the username and password.</p> <p>Depending on the authentication you want to use, set <code>emailService.graph.clientSecretCredentialFlow</code> to <code>true</code> or <code>false</code>. If <code>emailService.graph.clientSecretCredentialFlow=true</code>, <code>clientSecret</code> is used for authentication. If set to <code>false</code>, the username and password are used for authentication.</p> <hr/> <div style="display: flex; align-items: center;">  <ul style="list-style-type: none"> • For <code>clientsecretcredential</code> flow, <code>clientId</code>, <code>tenantid</code>, <code>clientsecret</code> and <code>userid</code> are </div> <hr/>

mandatory parameters.

- For `username` and `password` flow, `clientId`, `username` and `password` are mandatory parameters.

If any of the mandatory parameters are missed, a bean is created using the default values mentioned in the `js.quartz.properties` file.

`azureSql.resourceGroupName`

To fetch the list of databases, the resource group is required while creating `AzureSqlManagementService` bean. By default, it is set to `defaultResourceGroup`.

The bean for graph API is created only when the graph profile is enabled in `web.xml`. By default, the 'graph' profile in the `web.xml` file is disabled to prevent unnecessary bean creation. To enable the 'graph' profile:

```
<context-param>
<param-name>spring.profiles.active</param.name>
<param-value>default,engine,alerting-ie,jrs,graph</param.value>
</context-param>
```

Database Settings for the Quartz Driver Delegate Class

Quartz uses the Quartz driver delegate class to interact with the JDBC driver.



If you used `buildomatic` to install JasperReports Server, the correct value of the Quartz driver delegate class is automatically set for your database.

If you did not use buildomatic to install JasperReports Server, refer to the following table to edit the `js.quartz.properties` file and set the value of the Quartz driver delegate class to the correct value for your database.

Configuration File		
<code><app-server>/<deployment>/WEB-INF/js.quartz.properties</code>		
Property	Database	Value
<code>quartz.delegateClasses</code>	MySQL	<code>org.quartz.impl.jdbcjobstore.StdJDBCDelegate</code>
	PostgreSQL	<code>org.quartz.impl.jdbcjobstore.PostgreSQLDelegate</code>
	DB2	<code>org.quartz.impl.jdbcjobstore.DB2v8Delegate</code>
	Oracle	<code>org.quartz.impl.jdbcjobstore.StdJDBCDelegate</code>
	SQL Server ¹	<code>org.quartz.impl.jdbcjobstore.StdJDBCDelegate</code>
For SQL Server on WebSphere 8.5 use <code>org.quartz.impl.jdbcjobstore.MSSQLDelegate</code>		

Settings for the Report Scheduler Web URI

JasperReports Server uses the Report Scheduler Web URI to construct the link it sends in the output of a scheduled job. This link must be correct for the user to access the report on the server.

The port on which you run JasperReports Server and the context root of the deployed JasperReports Server web application determine the report scheduler Web URI. The default context root is `jasperserver`.

To set this value manually, edit this file:

```
<app-server>/<deployment>/WEB-INF/js.quartz.properties.
```

Change the properties as shown in the following table.

Property	App Server	Example Value
report.scheduler.web.deployment.uri	Apache Tomcat	http://localhost:8080/jasperserver-pro
	JBoss	http://localhost:8080/jasperserver-pro
	WebLogic	http://localhost:7001/jasperserver-pro
	WebSphere	http://localhost:9080/jasperserver-pro

Settings for the Quartz Table Prefix

For databases that support schemas, such as Oracle, SQL Server, and DB2, you can set the Quartz table prefix to include the schema, if you use one. In the default configuration, only DB2 requires an explicit schema name.



If you installed JasperReports Server using buildomatic, the Quartz table prefix is set automatically.

To set this value, edit the file `<app-server>/<deployment>/WEB-INF/js.quartz.properties`. Change the following property:

Property	Description
quartz.tablePrefix	The prefix for the quartz table, including any schema name, for example <code>JSPRSRVR.QRTZ_</code> for DB2.

Settings for Import-Export

If you manually configure the import-export shell scripts instead of using the buildomatic, make sure your settings for the Quartz driver delegate class property are correct for your database.



If you install using buildomatic, these settings are handled automatically (in buildomatic import-export).

To configure the import-export scripts manually, edit this file:

```
<js-install>/buildomatic/conf_source/iePro/js.quartz.properties
```

Change the following properties:

Property	Description
<code>quartz.delegateClass</code>	Set to the same value as described in Database Settings for the Quartz Driver Delegate Class .
<code>quartz.tablePrefix</code>	Set to the same value as described in Settings for the Quartz Table Prefix .

Setting Properties in the default_master.properties File

You can modify the `default_master.properties` file to configure the JasperReports Server functionality. Uncomment the properties that you want to have them take effect on installation. The properties are documented directly in the `default_master.properties` file:

```
<js-install>/buildomatic/default_master.properties
```

You will find a sample `master.properties` here (in the case of PostgreSQL):

```
<js-install>/buildomatic/sample_conf/postgresql_master.properties
```

When you run the `js-install.sh/bat` script (or the underlying `deploy-webapp-pro` ant target), these properties will be set in the deployed JasperReports Server in the `js.quartz.properties` file.

You can set the following properties in `default_master.properties` file (default values are shown):

- `notification.service.multiTenant.config=none`
- `emailService.type=mail`

- `calenderTrigger.resetStartTimeOnImport=false`
- `simpleTrigger.resetStartTimeOnImport=false`

Report Scheduler Email Properties

You can set the following properties to configure the Report Scheduler email (default values are shown):

```
quartz.mail.sender.host=mail.localhost.com
```

```
quartz.mail.sender.port=25
```

```
quartz.mail.sender.protocol=smtp
```

```
quartz.mail.sender.username=admin
```

```
quartz.mail.sender.password=password
```

```
quartz.mail.sender.from=admin@localhost.com
```

```
quartz.web.deployment.uri=http://localhost:8080/jasperserver-pro
```

For information about the Prerequisite for Azure Graph API login, see [Mail Server Configuration Settings Using Graph API](#).

You can set the following properties to configure the Report Scheduler email using a graph API (default values are shown):

```
emailService.graph.oauthUrl=https://graph.microsoft.com
```

```
emailService.graph.oauthClientId=oauthClientId
```

```
emailService.graph.oauthSecret=oauthSecret
```

```
emailService.graph.oauthTenantId=oauthTenantId
```

```
emailService.graph.oauthUserName=username
```

```
emailService.graph.oauthPassword=password
```

```
emailService.graph.oauthUserId=oauthUserId
```

```
emailService.graph.clientSecretCredentialFlow=true
```

```
azureSql.resourceGroupName=defaultResourceGroup
```



When sending a file with email:

- If the file size is under 3 MB, graph uses single POST.
-

-
- If the file size is in between 3MB and 150 MB it uses the largeAttachment upload flow.

For details, refer to [Outlook-Large-Attachments](#).

For information about properties to be set when composing email, refer to the *Setting Properties in the jasperserver_config.properties File* section in the JasperReports® Server Administrator Guide.

Diagnostic Properties

The following properties configure the Diagnostic functionality:

```
diagnostic.jmx.usePlatformServer = false
```

```
diagnostic.jmx.port = 10990
```

```
diagnostic.jmx.name = jasperserver
```

```
diagnostic.jmx.rmiHost = localhost
```

Look at the descriptions of the properties in the `default_master.properties` file and also refer to the *JasperReports Server Administrator Guide* for more information on these settings.

Settings to Skip Calendar Job Execution Immediately After Import

When importing scheduled jobs, all calendar jobs with misfired triggers are re-executed. This results in unwanted and unplanned executions. This can be overcome by changing the Misfire policy on the server. But, that will impact all other jobs along with the calendar trigger.

To avoid this, you can add two new properties `calendarTrigger.resetStartTimeOnImport` and `simpleTrigger.resetStartTimeOnImport` to stop executing calendar jobs immediately after import.

These properties, when set to `false`, skip executing calendar jobs right after import. The job is executed at the next trigger (calendar or simple). On setting the properties to `true`, the start time of the job is reset to the current time and it is not considered as misfired after the import.

Configuring Scheduler Misfire Policy

Property	Description
<code>calendarTrigger.resetStartTimeOnImport</code>	By default, <code>calendarTrigger.resetStartTimeOnImport=false</code> . If set to <code>true</code> , then while importing the job, start time is recalculated to the import time. And, the job execution starts in the future based on the condition set in calendar trigger.
<code>simpleTrigger.resetStartTimeOnImport</code>	By default, <code>simpleTrigger.resetStartTimeOnImport=false</code> . If set to <code>true</code> , then job start time is refreshed to current time.



For old jobs with simple triggers that were completed, but remained in the system, the jobs are executed again because the start time recalculated to the current time.

Updating XML/A Connection Definitions

Sample XML/A connections are included with the JasperReports Server sample data. If you plan to use XML/A Web Services in your environment, you may want to update the hard coded values in the sample connections.

If you have Jaspersoft OLAP enabled (via your license), JasperReports Server can make XML/A connections over the Web Services interface. These connections need a user account for authentication. You may have different usernames and passwords than the defaults in the sample data. Additionally, your application server hostnames and port may be different than the default values. In such cases, the connections and resources that rely on them will fail.

The sample connections are:

- Foodmart Sample XML/A connection
- SugarCRM Sample XML/A connection

To validate and update these resources:

1. Log into JasperReports Server as an administrator (like `jasperadmin`).
2. Navigate to the Repository Management page (View> Repository).

3. Click to expand the Analysis Components folder, then the Analysis Connections folder. Click to highlight Foodmart XML/A Connection, then click Edit.
4. Edit the following fields:
 - URI (hostname and port)
 - Login Username
 - Login Password
5. Click Next, then Save.
6. Make the same updates for SugarCRM XML/A Connection.

Installing the WAR File for WebSphere

JasperReports Server supports deployment on the IBM WebSphere Application Server, but requires its own database to store information such as users, organizations, and the repository. WebSphere users must use the WAR file distribution to install JasperReports Server. Download the WAR file distribution from [Jaspersoft Technical Support](https://www.jaspersoft.com/support) (<https://www.jaspersoft.com/support>) or contact your sales representative. The WAR file distribution comes in a file named `js-jrs_9.0.0_bin.zip`.

The WAR file distribution also includes two sample databases containing data for optional demos. For evaluation, we recommend installing the sample databases. In a production environment, we advise against installing sample data of any kind. You create and initialize the required repository database and the optional sample databases before JasperReports Server is deployed in WebSphere. The WebSphere administrator uses the WebSphere Administrative Console to deploy JasperReports Server.

This chapter contains the following sections:

- [Procedure for Installing and Deploying the WAR File in WebSphere](#)
- [Logging into the Server](#)
- [Configuring Report Scheduling](#)
- [Updating XML/A Connection Definitions \(Optional\)](#)
- [Troubleshooting your Configuration](#)

Procedure for Installing and Deploying the WAR File in WebSphere

Perform the procedures in this section to install and deploy the JasperReports Server WAR file in WebSphere.

Installing WebSphere and a Database

To install WebSphere and a database

1. Make sure you are using a supported version of WebSphere. See the TIBCO JasperReports Server Supported Platform Datasheet for more information.
2. Check that the WebSphere installation created a JAVA_HOME system environment variable. The variable needs to be set to the JAVA directory in the WebSphere installation.
3. Install the database (PostgreSQL, MySQL, Oracle, SQL Server, or DB2).



The target database can be on a remote server. WebSphere should reside on the local machine.

Preparing Server Files

To prepare JasperReports Server files

1. Unpack js-jrs_9.0.0_bin.zip to a top-level directory. Unpacking the ZIP file creates the directory js-jrs_pro_9.0.0_bin.zip.
2. (Required) Manually create and load the JasperReports Server database.
3. (Optional) Manually create and load the sample databases. See [Manually Creating the JasperReports Server Database](#) for instructions.
4. (Required) Manually import the default users and organization.
 - a. Copy the <dbType>_master.properties file for your database from sample_conf and paste it to buildomatic:
 - cd <js-install>/buildomatic
 - Copy from: <js-install>/buildomatic/sample_conf
 - Paste to: <js-install>/buildomaticFor example, copy sample_conf/postgresql_master.properties to buildomatic.

- b. Rename the file that you copied to `default_master.properties`.
- c. Edit the `default_master.properties` file:
 - Set `appServerType` to `skipAppServerCheck`.
 - Change `dbUsername`, `dbPassword`, and `dbHost` to the appropriate settings for your database.
 - If you are using a port other than the default for your database, or if you have installed the database on a remote machine, change the `dbPort` field under Custom Properties to the appropriate settings.

For the Split installation, configure the additional settings in the `default_master.properties` file as described in [Additional Buildomatic Configuration for Split Installation](#).

For the JNDI restricted access installation, configure the additional settings in the `default_master.properties` file as described in [Installation Types](#).

Each `sample_conf/<dbType>_master.properties` file contains appropriate sample values.

- d. Start your database server.
- e. Open a Command Prompt as Administrator and run these commands:

Buildomatic Targets to Execute

Commands	Description
<code>cd <js-install>/buildomatic</code>	Go to the buildomatic directory.
<code>js-ant create-js-db</code>	Create the jasperserver repository database.
<code>js-ant init-js-db-pro</code> <code>js-ant import-minimal-pro</code>	Initializes database, loads core application data.
<code>js-ant create-sugarcrm-db</code> <code>js-ant create-foodmart-db</code>	(Optional) Creates sample databases.

Commands	Description
js-ant load-sugarcrm-db js-ant load-foodmart-db	(Optional) Loads sample data into the sample databases.
js-ant import-sample-data-pro	(Optional) Loads the demos that use the sample data.
js-ant create-audit-db	(Optional) Creates the audit database. Required only for the Split installation.
js-ant init-audit-db-pro	(Optional) Initializes the audit database. Required only for the Split installation.

5. Set up your license file. For more information, refer to [Setting Up the JasperReports Server License](#).



Installing JasperReports Server automatically generates encryption keys that reside on the file system. These keys are stored in a dedicated Jaspersoft keystore. Make sure that this keystore is properly secured and backed up, as described in the *JasperReports Server Security Guide*.

Configuring CSRFGuard, Hibernate, Quartz, and WebSphere Settings

Before deploying the JasperReports Server WAR file, update the CSRFGuard, Hibernate, Quartz, and settings as described here.

Configure CSRFGuard, Hibernate, and Quartz settings in the WAR file

1. The WAR file is an archive format in a single file.
 - a. Extract the `WebSphere.jrs.csrfguard.properties` file using the following command:

```
cd <js-install>
"%JAVA_HOME%/bin/jar" xf jasperserver-pro.war WEB-
INF/csrf/WebSphere.jrs.csrfguard.properties
```

This creates the WEB-INF/csrf folder in the current location and places the extracted file there.

- b. Rename the file from WebSphere.jrs.csrfguard.properties to jrs.csrfguard.properties using the following command:

```
mv ./WEB-INF/csrf/WebSphere.jrs.csrfguard.properties ./WEB-
INF/csrf/jrs.csrfguard.properties
```

2. Extract and rename the web-version24.xml file using the commands below:

```
cd <js-install>
"%JAVA_HOME%/bin/jar" xf jasperserver-pro.war WEB-INF/web-version24.xml
mv ./WEB-INF/web-version24.xml ./WEB-INF/web.xml
```

The jar command creates the WEB-INF folder in the current location and places the extracted file there.

Open the WEB-INF/web.xml file for editing and replace every occurrence of

```
<res-type>javax.sql.ConnectionPoolDataSource</res-type>
```

With

```
<res-type>javax.sql.DataSource</res-type>
```

For example, change the following:

```
<resource-ref>
  <description>JasperServer Metadata repository</description>
  <res-ref-name>jdbc/jasperserver</res-ref-name>
  <res-type>javax.sql.ConnectionPoolDataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

To:

```
<resource-ref>
  <description>JasperServer Metadata repository</description>
  <res-ref-name>jdbc/jasperserver</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Do the same for the Supermart, Foodmart, Audit, System Analytics and Audit Analytics databases.

3. Enable Hibernate persistence:

- a. Extract the applicationContext.xml file using the commands below:

```
cd <js-install>
"%JAVA_HOME%/bin/jar" xf jasperserver-pro.war WEB-
INF/applicationContext.xml
```

- b. Uncomment the following line:

```
<!--Hibernate-Validator Websphere fix-->
```

```
<bean id="HibernatePersistenceResolverWebsphereFix"
class="com.jaspersoft.hibernate.resolver.HibernatePersistenceProviderResol
ver"
init-method="register"/>
```

- c. In the same file, find the profileAttributesResolver bean:

```
<bean id="profileAttributesResolver"
```

```
class="com.jaspersoft.jasperserver.api.metadata.user.service.impl.ProfileAttributesResolverImpl">
```

Add `depends-on="HibernatePersistenceResolverWebSphereFix` to it:

```
<bean id="profileAttributesResolver"
class="com.jaspersoft.jasperserver.api.metadata.user.service.impl.ProfileAttributesResolverImpl"
depends-on="HibernatePersistenceResolverWebSphereFix">
```

- Copy the already configured files for `hibernate.properties` and `js.quartz.properties` to the `WEB-INF` folder.

(Buildomatic configured these files for your database type in the steps above.)

From:

```
<js-install>/buildomatic/build_conf/default/webapp/WEB-INF/classes/hibernate.properties
```

```
<js-install>/buildomatic/build_conf/default/webapp/WEB-INF/js.quartz.properties
```

To:

```
<js-install>/WEB-INF/classes
```

Copy the `../buildomatic/keystore.init.properties` file needs to the `../WEB-INF/classes` directory.

- Edit the scheduler URI port value for WebSphere in the `js.quartz.properties`:

Edit `js.quartz.properties`:

Set:

```
report.scheduler.web.deployment.uri=http://localhost:8080/jasperserver-pro
```

To:

```
report.scheduler.web.deployment.uri=http://localhost:9080/jasperserver-pro
```

- If you want to configure JasperReports Server to automatically schedule and email reports, enter your mail server information in the `js.quartz.properties` file. Modify all `report.scheduler.mail.sender.*` properties for your mail server.

7. Now that you have modified/updated the individual configuration files, you must replace them in the WAR file archive using the following commands.

```
cd <js-install>
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/classes/hibernate.properties
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/js.quartz.properties
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/csrf/jrs.csrfguard.properties
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/applicationContext.xml
```

8. If you have modified the web.xml file, replace that file in the WAR file archive using the following additional commands.

```
cd <js-install>
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/web.xml
```

9. Enter the following additional commands:

```
cd <js-install>
zip -d jasperserver-pro.war WEB-INF/lib/stax-api-1.0.2.jar
zip -d jasperserver-pro.war WEB-INF/lib/xercesImpl-2.12.0.jar
zip -d jasperserver-pro.war WEB-INF/lib/jta-1.1.jar
zip -d jasperserver-pro.war WEB-INF/lib/xml-apis-1.4.01.jar
zip -d jasperserver-pro.war WEB-INF/lib/javax.el-2.2.4.jar
zip -d jasperserver-pro.war WEB-INF/lib/javax.el-api-2.2.4.jar
zip -d jasperserver-pro.war WEB-INF/lib/batik\*
```

10. Add the latest batik-all.jar from Websphere:

- a. Find the jar using the following command:

```
find /opt/IBM/ -name "batik*.*"
```

- b. Copy the file into a lib directory in the jasperserver-pro.war file, for example:

```
cd <js-install>
mkdir -p WEB-INF/lib
cp /opt/IBM/WebSphere/AppServer/systemApps/isclite.ear/lib/batik-all.jar
WEB-INF/lib
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/lib/batik-all.jar
```

11. Clean up your file system by deleting the WEB-INF directory that you created along with the edited files it contains.

Configuring a JDBC Provider in WebSphere

To configure a JDBC Provider in WebSphere

1. Launch the WebSphere Administrative Console and navigate to Resources > JDBC > JDBC Providers.
2. On the JDBC providers page, click the Guided Activity link at the top of the JDBC Providers page and follow the Integrated Solutions Console instructions:
 - a. Configure credentials for a secure database:
 - Use the J2C authentication aliases panel to create a new authenticated user.
 - In Global Security, click New and enter the user alias, user ID, and password. The following table shows the credentials that WebSphere uses to access the database.

J2C Authentication Alias Settings

	Alias	Example User ID	Example Password
PostgreSQL	postgresql_jasperdb	postgres	postgres
MySQL	mysql_jasperdb	root	password
Oracle	jasperserver_user	jasperserver	password
DB2	db2admin_user	db2inst1	password
SQL Server	jasperserver_user	sa	sa

- b. Connect to a database panel. From the Scope dropdown, choose Node:<node_name>,Server=<server_name>
- c. Click the New button to create a new JDBC Provider.
- d. Select your database type:
 - If you are using PostgreSQL or MySQL — select User-defined.
 - If you want to use the JDBC driver built and distributed by the MySQL project, see [MySQL Example](#)
 - If you are using the DB2, Oracle, or SQL Server — select the appropriate database.
- e. Select or enter these options. Your options depend on your database type.

Database type	Implementation class name or type	Name
User-defined (PostgreSQL)	org.postgresql.jdbc2.optional.ConnectionPool	PostgreSQL JDBC Provider
User-defined (MySQL)	org.mariadb.jdbc.MySQLDataSource	MySQL JDBC Provider
User-defined (MySQL)	com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource	MySQL JDBC Provider
DB2 Universal JDBC Driver Provider	Connection pool data source	DB2 Universal JDBC Driver Provider
Oracle	Connection pool data source	Oracle JDBC Driver
SQL Server	Connection pool data source	Microsoft SQL Server JDBC Driver

3. Click Next and enter the database classpath information for the JDBC provider.

- a. For PostgreSQL, MySQL, Oracle, SQL Server, and DB2, enter the following:

```
<js-install>\builddomatic\conf_source\db\<your_database>\jdbc\
```

For example, enter:

```
C:\js-jrs_9.0.0_bin\builddomatic\conf_source\db\postgresql\jdbc\postgresql-42.2.5.jar
```

Alternatively, you can copy the jar to a location in your WebSphere deployment and specify that location for the JDBC driver path.



If JasperReports Server is deployed on the same host as DB2, delete the following file to avoid conflicts: `<db2>/SQLLIB/java/db2jcc.jar`

4. To ensure you have full support for import/export from the command line, copy your JDBC driver to the following location. If you are not using the command line for import/export, you can skip this step:

from: `<js-install>\buildomatic\conf_source\db\<your_database>\jdbc\`

to: `<js-install>\buildomatic\conf_source\iePro\lib`

5. Click Next to proceed to the next step.
6. Review the JDBC provider information that you entered and click Finish.

To define the jasperserver JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, for PostgreSQL, click PostgreSQL JDBC Provider.



To use a database other than PostgreSQL, configure the database connections and custom properties as described in [Configuring Other Database Connections](#).

2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jasperserver`
5. Enter the JNDI name: `jdbc/jasperserver`
6. Click Next, choose Select an existing JDBC provider, then select PostgreSQL JDBC Provider from the dropdown list.
7. Click Next and accept the default helper class (`com.ibm.websphere.rsadapter.GenericDataStoreHelper`). Select the checkbox to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases, as shown in the following table.

Field Name	PostgreSQL Value
Component-managed authentication alias	postgresql_jasperdb
Mapping configuration alias	DefaultPrincipalMapping
Container-managed authentication alias	postgresql_jasperdb

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jasperserver data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the check box for the newly created jasperserver data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jasperserver data sources General Properties page.
3. In Additional Properties, on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select databaseName. Set the value to jasperserver.
5. Set serverName to the correct value for your server.

To define the jsSystemAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, for PostgreSQL, click PostgreSQL JDBC Provider.



To use a database other than PostgreSQL, configure the database connections and custom properties as described in [Configuring Other Database Connections](#).

2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsSystemAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverSystemAnalytics`
6. Click Next, choose Select an existing JDBC provider, then select PostgreSQL JDBC Provider from the dropdown list.
7. Click Next and accept the default helper class (`com.ibm.websphere.rsadapter.GenericDataStoreHelper`). Select the checkbox to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases, as shown in the following table.

Field Name	PostgreSQL Value
Component-managed authentication alias	postgresql_jasperdb
Mapping configuration alias	DefaultPrincipalMapping
Container-managed authentication alias	postgresql_jasperdb

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created `jsSystemAnalytics` data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.

4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the check box for the newly created `jsSystemAnalytics` data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jasperserver data sources General Properties page.
3. In Additional Properties, on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select `databaseName`. Set the value to `jsSystemAnalytics`.
5. Set `serverName` to the correct value for your server.

To define the jsaudit JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, for PostgreSQL, click PostgreSQL JDBC Provider.



To use a database other than PostgreSQL, configure the database connections and custom properties as described in [Configuring Other Database Connections](#).

2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsaudit`
5. Enter the JNDI name: `jdbc/jasperserverAudit`
6. Click Next, choose Select an existing JDBC provider, then select PostgreSQL JDBC Provider from the drop-down list.
7. Click Next and accept the default helper class (`com.ibm.websphere.rsadapter.GenericDataStoreHelper`). Select the check box to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases, as shown in the following table.

Field Name	PostgreSQL Value
Component-managed authentication alias	postgresql_jasperdb
Mapping configuration alias	DefaultPrincipalMapping
Container-managed authentication alias	postgresql_jasperdb

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsaudit data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the check box for the newly created jsaudit data source and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

2. Navigate to the jsaudit data sources General Properties page.
3. In Additional Properties, on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select databaseName. Set the value to:
 - For Compact installation: jasperserver
 - For Split installation: jsaudit
5. Set serverName to the correct value for your server.

To create optional sugarcrm and foodmart data sources

1. If you plan to run the sample reports, use the values in the following table to create the foodmart and sugarcrm JNDI data sources.

Field Name	Value	
Data source name	foodmart	sugarcrm
JNDI name	jdbc/foodmart	jdbc/sugarcrm

2. Click Save directly to the master configuration.

Next, deploy the WAR file in WebSphere as described in [Deploying the WAR File in WebSphere](#).

To define the jsAuditAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, for PostgreSQL, click PostgreSQL JDBC Provider.



To use a database other than PostgreSQL, configure the database connections and custom properties as described in [Configuring Other Database Connections](#).

2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jsAuditAnalytics
5. Enter the JNDI name: jdbc/jasperserverAuditAnalytics
6. Click Next, choose Select an existing JDBC provider, then select PostgreSQL JDBC Provider from the drop-down list.
7. Click Next and accept the default helper class (com.ibm.websphere.rsadapter.GenericDataStoreHelper). Select the check box to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases, as shown in the following table.

Field Name	PostgreSQL Value
Component-managed authentication alias	postgresql_jasperdb
Mapping configuration alias	DefaultPrincipalMapping
Container-managed authentication alias	postgresql_jasperdb

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsaudit data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the check box for the newly created jsAuditAnalytics data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jsaudit data sources General Properties page.
3. In Additional Properties, on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select databaseName. Set the value to:
 - For Compact installation: jasperserver
 - For Split installation: jsaudit
5. Set serverName to the correct value for your server.

Deploying the WAR File in WebSphere

To deploy the JasperReports Server WAR file in WebSphere

1. In the Administrative Console, navigate to Applications > New Application and select New Enterprise Application.

JasperReports Server is a modern application, based on Java Servlet version 2.4, so you do not select the older, WebSphere V4-compliant application type.

2. Browse to `<js-install>/jasperserver-pro.war` on the local file system. Keep the default setting (Fast path) selected and click Next.
3. On the Select installation options page, accept all the default settings and click Next.
4. On the Map modules to servers page, make sure the JasperReports Server module is mapped to the cell, node, and server that you want. Click Next.
5. On the Map modules to servers page, select `jasperserver`. Click Next.
6. On the Map resource references to resources page, map the resources you want:
 - a. First, select the Browse button under the `jdbc/jasperserver` resource. In the page that opens, select the `jdbc/jasperserver` radio button, and click Apply. Then select the check box next to the `jdbc/jasperserver` resource.
 - b. Select the Browse button under the `jdbc/jasperserverAudit` resource. In the page that opens, select the `jdbc/jasperserverAudit` radio button, and click Apply. Then select the check box next to the `jdbc/jasperserverAudit` resource.
 - c. Select the **Browse** button under the `jdbc/jasperserverSystemAnalytics` resource. In the page that opens, select the `jdbc/jasperserverSystemAnalytics` radio button, and click **Apply**. Then select the check box next to the `jdbc/jasperserverSystemAnalytics` resource.
 - d. Select the **Browse** button under the `jdbc/jasperserverAuditAnalytics` resource. In the page that opens, select the `jdbc/jasperserverAuditAnalytics` radio button, and click **Apply**. Then select the check box next to the `jdbc/jasperserverAuditAnalytics` resource.
 - e. If you plan to run the sample reports, follow the same steps for `jdbc/surgarcrm` and `jdbc/foodmart`, making sure to select the correct radio button for each one.
 - f. When you have mapped all resources, select the check boxes next to every resource have mapped.
 - g. Click Next.

7. On the Map virtual hosts page, choose the JasperServer UI application module. Click Next.
8. In the Map context roots for Web modules, enter `jasperserver-pro`.
9. Click Next, review the summary information and start the installation process. (The installation process may take a while.)
10. Click Save directly to the master configuration.

Setting JVM Options

To set the Java JVM Options

For the JasperReports Server XML/A functionality to work, special Java JVM options need to be set to resolve class conflicts between the WebSphere and JasperReports Server web services implementation. JVM options also provide the optimal resources for running JasperReports Server.

To configure your Java JVM options

1. Select Enterprise Applications > `jasperserver-pro_war` > Target specific application status > (server name).
2. Expand Java and Process Management > Process Definition > Java Virtual Machine > Generic JVM arguments.
3. In the Generic JVM Options text box, paste in the following JVM options that explicitly specify JasperReports Server classes for Xalan, as well as optimize JVM resources. The memory settings are a recommended minimum; you may need to increase the memory assigned to the JVM according to your usage:

Generic JVM Options on Windows

Options for all databases	<code>-Dclient.encoding.override=UTF-8 -Xms2048m -Xmx4096m -Xss2m -XX:+UseG1GC -Dlog4j.configurationFile=WEB-INF/log4j2.properties</code>
---------------------------	---

Additional option for Java	<code>-Djava.locale.providers=COMPAT</code>
----------------------------	---

Generic JVM Options on Windows

11

Additional option for Oracle	<code>-Doracle.jdbc.defaultNChar=true</code>
------------------------------	--

Additional option for FTPS connections	<code>-Dcom.ibm.jsse2.overrideDefaultTLS=true</code>
--	--

Generic JVM Options on Linux

Options for all databases	<code>-Dclient.encoding.override=UTF-8 -Xms2048m -Xmx4096m -Xss2m -XX:+UseG1GC -Dlog4j.configurationFile=WEB-INF/log4j2.properties</code>
---------------------------	---

Additional option for Java 11	<code>-Djava.locale.providers=COMPAT</code>
-------------------------------	---

Additional option for Oracle	<code>-Doracle.jdbc.defaultNChar=true</code>
------------------------------	--

Additional option for FTPS connections	<code>-Dcom.ibm.jsse2.overrideDefaultTLS=true</code>
--	--



Setting the Oracle localization option, `defaultNChar`, can substantially impact the performance of JDBC queries. If you don't need to support UTF-8 for your Oracle database, you can omit this setting.

4. Click Save on the console task bar.

To configure class loading

1. Select Enterprise Applications > jasperserver-pro_war > Class loading and update detection.
2. In the section Class loader order, select Classes loaded with local class loader first (parent last).
3. In the WAR class loader policy section select Single class loader for application.
4. Save directly to your master configuration.
5. Restart WebSphere.

Next, start JasperReports Server as described in [Starting and Restarting JasperReports Server](#).

Configuring Other Database Connections

Defining a JNDI Name and Sample Data Sources for MySQL

To define the jasperserver JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, MySQL JDBC Provider.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jasperserver
5. Enter the JNDI name: jdbc/jasperserver
6. Click Next, choose Select an existing JDBC provider, then select MySQL JDBC Provider from the dropdown list.
7. Click Next and accept the default helper class (com.ibm.websphere.rsadapter.GenericDataStoreHelper). Select the checkbox to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases:

Field Name	MySQL Value
Component-managed authentication alias	mysql_jasperdb
Mapping configuration alias	DefaultPrincipalMapping
Container-managed authentication alias	mysql_jasperdb

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jasperserver data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jasperserver data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jasperserver data sources General Properties page.
3. In Additional Properties on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select databaseName. Set the value to jasperserver.
5. Create a new property called url. Enter the following value and save the change:
`jdbc:mysql://localhost/jasperserver?useUnicode=true&characterEncoding=UTF-8&tinyInt1isBit=false&allowPublicKeyRetrieval=true`
6. Click Save directly to the master configuration.

To define the jsSystemAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, MySQL JDBC Provider.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsSystemAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverSystemAnalytics`
6. Click Next, choose Select an existing JDBC provider, then select MySQL JDBC Provider from the dropdown list.
7. Click Next and accept the default helper class (`com.ibm.websphere.rsadapter.GenericDataStoreHelper`). Select the checkbox to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases:

Field Name	MySQL Value
Component-managed authentication alias	<code>mysql_jasperdb</code>
Mapping configuration alias	<code>DefaultPrincipalMapping</code>
Container-managed authentication alias	<code>mysql_jasperdb</code>

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created `jsSystemAnalytics` data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsSystemAnalytics data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jasperserver data sources General Properties page.
3. In Additional Properties on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select databaseName. Set the value to jasperserver.
5. Create a new property called url. Enter the following value and save the change:
`jdbc:mysql://localhost/jasperserver?useUnicode=true&characterEncoding=UTF-8&tinyInt1isBit=false&allowPublicKeyRetrieval=true`
6. Click Save directly to the master configuration.

To define the jsaudit JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, MySQL JDBC Provider.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jsaudit
5. Enter the JNDI name: jdbc/jasperserverAudit
6. Click Next, choose Select an existing JDBC provider, then select MySQL JDBC Provider from the dropdown list.
7. Click Next and accept the default helper class (com.ibm.websphere.rsadapter.GenericDataStoreHelper). Select the checkbox to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases:

Field Name	MySQL Value
Component-managed authentication alias	mysql_jasperdb
Mapping configuration alias	DefaultPrincipalMapping
Container-managed authentication alias	mysql_jasperdb

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsaudit data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsaudit data source and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

2. Navigate to the jsaudit data sources General Properties page.
3. In Additional Properties on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select `databaseName`. Set the value to
 - For Compact installation: `jasperserver`
 - For Split installation: `jsaudit`

5. Create a new property called url. Enter the following value and save the change:
 - For Compact installation:
`jdbc:mysql://localhost/jasperserver?useUnicode=true&characterEncoding=UTF-8&tinyInt1isBit=false&allowPublicKeyRetrieval=true`
 - For Split installation:
`jdbc:mysql://localhost/jsaudit?useUnicode=true&characterEncoding=UTF-8&tinyInt1isBit=false&allowPublicKeyRetrieval=true`
6. Click Save directly to the master configuration.

To define the jsAuditAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider that you just created. For example, MySQL JDBC Provider.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsAuditAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverAuditAnalytics`
6. Click Next, choose Select an existing JDBC provider, then select MySQL JDBC Provider from the dropdown list.
7. Click Next and accept the default helper class (`com.ibm.websphere.rsadapter.GenericDataStoreHelper`). Select the checkbox to use this data source in container managed persistence (CMP).
8. Click Next and select the Setup security aliases:

Field Name	MySQL Value
Component-managed authentication alias	<code>mysql_jasperdb</code>
Mapping configuration alias	<code>DefaultPrincipalMapping</code>
Container-managed authentication alias	<code>mysql_jasperdb</code>

9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsAuditAnalytics data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsAuditAnalytics data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jsaudit data sources General Properties page.
3. In Additional Properties on the right side of the General Properties page, click Custom properties.
4. Scroll down the list of properties and select databaseName. Set the value to
 - For Compact installation: `jasperserver`
 - For Split installation: `jsaudit`
5. Create a new property called url. Enter the following value and save the change:
 - For Compact installation:
`jdbc:mysql://localhost/jasperserver?useUnicode=true&characterEncoding=UTF-8&tinyInt1isBit=false&allowPublicKeyRetrieval=true`
 - For Split installation:
`jdbc:mysql://localhost/jsaudit?useUnicode=true&characterEncoding=UTF-8&tinyInt1isBit=false&allowPublicKeyRetrieval=true`
6. Click Save directly to the master configuration.

To create optional sugarcrm and foodmart data sources

1. If you plan to run the sample reports, use the values in the following table to create the foodmart and sugarcrm JNDI data sources.

Field Name	Value	
Data source name	foodmart	sugarcrm
JNDI name	jdbc/foodmart	jdbc/sugarcrm

2. Click Save directly to the master configuration.
3. Set the connection pool size as described in [To set the connection pool size](#).

Next, deploy the WAR file in WebSphere as described in [Deploying the WAR File in WebSphere](#).

Defining a JNDI Name and Sample Data Sources for DB2

To define the JSPRSRVR JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, DB2 Universal JDBC Provider.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: JSPRSRVR
5. Enter the JNDI name: jdbc/jasperserver
6. Click Next.
7. For DB2 driver, choose Select an existing JDBC provider, then select DB2 Universal JDBC Provider from the dropdown list.

8. Click Next and enter these values:

Field Name	Value
Driver type	4
Database name	JSPRSRVR
Server name	localhost
Port number	50000

9. Select Use this data source in CMP and click Next.

10. On the Setup security aliases page, enter the following value for the Component-managed authentication alias:

db2admin_user

11. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created **JSPRSRVR** data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created JSPRSRVR data source, and click Test Connection.
2. In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

3. Edit the following properties, adding any that are missing, then save the changes:

Properties for DB2 JDBC Driver

Property Name	Value
currentSchema	JSPRSRVR
fullyMaterializeLobData	true
fullyMaterializeInputStreams	true
progressiveStreaming	2
progressiveLocators	2

4. Go back to the list of JDBC data sources, select the checkbox for the JSPRSRVR data source, and click Test Connection.

To define the jsSystemAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, DB2 Universal JDBC Provider.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsSystemAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverSystemAnalytics`
6. Click Next.
7. For DB2 driver, choose Select an existing JDBC provider, then select DB2 Universal JDBC Provider from the dropdown list.

- Click Next and enter these values:

Field Name	Value
Driver type	4
Database name	JSPRSRVR
Server name	localhost
Port number	50000

- Select Use this data source in CMP and click Next.
- On the Setup security aliases page, enter the following value for the Component-managed authentication alias:
db2admin_user
- Click Next, review the summary information, and click Finish.

To set the connection pool size

- In the list of JDBC data sources, click the newly created **jsSystemAnalytics** data source to edit it.
- Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
- Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
- Click Save.

To define custom properties

- In the list of JDBC data sources, select the checkbox for the newly created jsSystemAnalytics data source, and click Test Connection.
- In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

3. Edit the following properties, adding any that are missing, then save the changes:

Properties for DB2 JDBC Driver

Property Name	Value
currentSchema	JSPRSRVR
fullyMaterializeLobData	true
fullyMaterializeInputStreams	true
progressiveStreaming	2
progressiveLocators	2

4. Go back to the list of JDBC data sources, select the checkbox for the jsSystemAnalytics data source, and click Test Connection.

To define the jsaudit JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, DB2 Universal JDBC Provider.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jsaudit
5. Enter the JNDI name: jdbc/jasperserverAudit
6. Click Next.
7. For DB2 driver, choose Select an existing JDBC provider, then select DB2 Universal JDBC Provider from the dropdown list.
8. Click Next and enter these values:

Field Name	Value
Driver type	4

Field Name	Value
Database name	For Compact installation: JSPRSRVR For Split installation: JSAUDIT
Server name	localhost
Port number	50000

9. Select Use this data source in CMP and click Next.
10. On the Setup security aliases page, enter the following value for the Component-managed authentication alias:
db2admin_user
11. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsaudit data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsaudit data source, and click Test Connection.
2. In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

3. Edit the following properties, adding any that are missing, then save the changes:

Properties for DB2 JDBC Driver

Property Name	Value
currentSchema	For Compact Installation: JSPRSVR For Split Installation: JSAUDIT
fullyMaterializeLobData	true
fullyMaterializeInputStreams	true
progressiveStreaming	2
progressiveLocators	2

4. Go back to the list of JDBC data sources, select the checkbox for the jsaudit data source, and click Test Connection.

To define the jsAuditAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, DB2 Universal JDBC Provider.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsAuditAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverAuditAnalytics`
6. Click Next.
7. For DB2 driver, choose Select an existing JDBC provider, then select DB2 Universal JDBC Provider from the dropdown list.

8. Click Next and enter these values:

Field Name	Value
Driver type	4
Database name	For Compact installation: JSPRSRVR For Split installation: JSAUDIT
Server name	localhost
Port number	50000

9. Select Use this data source in CMP and click Next.

10. On the Setup security aliases page, enter the following value for the Component-managed authentication alias:

db2admin_user

11. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsAuditAnalytics data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsAuditAnalytics data source, and click Test Connection.
2. In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

3. Edit the following properties, adding any that are missing, then save the changes:

Properties for DB2 JDBC Driver

Property Name	Value
currentSchema	For Compact Installation: JSPRSVR For Split Installation: JSAUDIT
fullyMaterializeLobData	true
fullyMaterializeInputStreams	true
progressiveStreaming	2
progressiveLocators	2

4. Go back to the list of JDBC data sources, select the checkbox for the jsAuditAnalytics data source, and click Test Connection.

To create optional sugarcrm and foodmart data sources

1. If you plan to run the sample reports, use the following values to create the foodmart and sugarcrm JNDI data sources.

Field Values for Optional Data Sources for DB2 Drivers with WebSphere

Field Name	Value	
Data source name	foodmart	sugarcrm
JNDI name	jdbc/foodmart	jdbc/sugarcrm
Component-managed authentication alias	<node>/db2admin_user	
Database name	foodmart	sugarcrm
Driver type	4	

Field Name	Value
Server name	localhost
Port number	50000
Use this data source in CMP	selected

Custom Properties for DB2 Driver with WebSphere

Property Name	Value
currentSchema	FOODMART SUGARCRM
resultSetHoldability	1

2. Click Save directly to the master configuration.
3. Set the connection pool size as described in [To set the connection pool size](#).

Next, deploy the WAR file in WebSphere as described in [Deploying the WAR File in WebSphere](#).

Defining a JNDI Name and Sample Data Sources for Oracle

To define the jasperserver JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, Oracle JDBC Driver.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jasperserver
5. Enter the JNDI name: jdbc/jasperserver
6. Click Next.

7. For Oracle driver, choose Select an existing JDBC provider, then select Oracle JDBC Driver from the dropdown list.
8. Click Next and enter the following values:

Field Name	Value
URL	jdbc:oracle:thin:@localhost:1521:orcl
Data store helper class name	Oracle11g data store helper
Use this data source in CMP	selected

9. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
 - `jasperserver_user`
10. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jasperserver data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jasperserver data source and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jasperserver data sources General Properties page.
3. In Additional Properties on the right side, click Custom properties.

4. Go back to the list of JDBC data sources, select the checkbox for the jasperserver data source, and click Test Connection.

To define the jsSystemAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, Oracle JDBC Driver.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsSystemAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverSystemAnalytics`
6. Click Next.
7. For Oracle driver, choose Select an existing JDBC provider, then select Oracle JDBC Driver from the dropdown list.
8. Click Next and enter the following values:

Field Name	Value
URL	<code>jdbc:oracle:thin:@localhost:1521:orcl</code>
Data store helper class name	Oracle11g data store helper
Use this data source in CMP	selected

9. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
`jasperserver_user`
10. Click Next, review the summary information and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsSystemAnalytics data source to edit it.

2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsSystemAnalytics data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jasperserver data sources General Properties page.
3. In Additional Properties on the right side, click Custom properties.
4. Go back to the list of JDBC data sources, select the checkbox for the jsSystemAnalytics data source, and click Test Connection.

To define the jsaudit JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, Oracle JDBC Driver.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jsaudit
5. Enter the JNDI name: jdbc/jasperserverAudit
6. Click Next.
7. For Oracle driver, choose Select an existing JDBC provider, then select Oracle JDBC Driver from the dropdown list.
8. Click Next and enter the following values:

Field Name	Value
URL	jdbc:oracle:thin:@localhost:1521:orcl
Data store helper class name	Oracle11g data store helper
Use this data source in CMP	selected

- Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
`jasperserver_user`
- Click Next, review the summary information and click Finish.

To set the connection pool size

- In the list of JDBC data sources, click the newly created jsaudit data source to edit it.
- Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
- Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
- Click Save.

To define custom properties

- In the list of JDBC data sources, select the checkbox for the newly created jsaudit data source and click Test Connection.
 In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
- Navigate to the jsaudit data sources General Properties page.
- In Additional Properties on the right side, click Custom properties.
- Go back to the list of JDBC data sources, select the checkbox for the jsaudit data source, and click Test Connection.

To define the jsAuditAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, Oracle JDBC Driver.
2. Click Data sources in the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsAuditAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverAuditAnalytics`
6. Click Next.
7. For Oracle driver, choose Select an existing JDBC provider, then select Oracle JDBC Driver from the dropdown list.
8. Click Next and enter the following values:

Field Name	Value
URL	<code>jdbc:oracle:thin:@localhost:1521:orcl</code>
Data store helper class name	Oracle11g data store helper
Use this data source in CMP	selected

9. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
 - `jasperserver_user`
10. Click Next, review the summary information and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created `jsAuditAnalytics` data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsAuditAnalytics data source and click Test Connection.
In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.
2. Navigate to the jsaudit data sources General Properties page.
3. In Additional Properties on the right side, click Custom properties.
4. Go back to the list of JDBC data sources, select the checkbox for the jsAuditAnalytics data source, and click Test Connection.

To create optional sugarcrm and foodmart data sources

1. If you plan to run the sample reports, use the following values to create the foodmart and sugarcrm JNDI data sources:

Field Name	Value	
Data source name	foodmart	sugarcrm
JNDI name	jdbc/foodmart	jdbc/sugarcrm
Component-managed authentication alias	<node>/foodmart_user	<node>/sugarcrm_user

Property Name	Value
portNumber	1521
serverName	localhost

2. Click Save directly to the master configuration.
3. Set the connection pool size as described in [To set the connection pool size](#).

Next, deploy the WAR file in WebSphere as described in [Deploying the WAR File in WebSphere](#).

Defining a JNDI Name and Sample Data Sources for SQL Server

To define the jasperserver JDBC provider

1. Click the name of the JDBC provider you just created. For example, Microsoft SQL Server JDBC Driver.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jasperserver`
5. Enter the JNDI name: `jdbc/jasperserver`
6. Click Next.
7. For SQL Server driver, choose Select an existing JDBC provider, then select Microsoft SQL Server JDBC Driver from the dropdown list.
8. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
`jasperserver_user`
9. Click Next, review the summary information and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jasperserver data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jasperserver data source, and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

2. Navigate to the jasperserver data sources General Properties page.

To define the jsSystemAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, Microsoft SQL Server JDBC Driver.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsSystemAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverSystemAnalytics`
6. Click Next.
7. For SQL Server driver, choose Select an existing JDBC provider, then select Microsoft SQL Server JDBC Driver from the dropdown list.
8. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
`jasperserver_user`
9. Click Next, review the summary information and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsSystemAnalytics data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsSystemAnalytics data source, and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

2. Navigate to the jasperserver data sources General Properties page.

To define the jsaudit JDBC provider

1. Click the name of the JDBC provider you just created. For example, Microsoft SQL Server JDBC Driver.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: jsaudit
5. Enter the JNDI name: jdbc/jasperserverAudit
6. Click Next.
7. For SQL Server driver, choose Select an existing JDBC provider, then select Microsoft SQL Server JDBC Driver from the dropdown list.
8. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
`jasperserver_user`
9. Click Next, review the summary information and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsaudit data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsaudit data source, and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

2. Navigate to the jsaudit data sources General Properties page.

To define the jsAuditAnalytics JDBC data source and expose it through JNDI

1. Click the name of the JDBC provider you just created. For example, Microsoft SQL Server JDBC Driver.
2. Click Data sources under the Additional Properties of the JDBC provider details panel.
3. To create a new data source, click New. The new data source wizard appears.
4. Enter the data source name: `jsAuditAnalytics`
5. Enter the JNDI name: `jdbc/jasperserverAuditAnalytics`
6. Click Next.
7. For SQL Server driver, choose Select an existing JDBC provider, then select Microsoft SQL Server JDBC Driver from the dropdown list.
8. Click Next and in Setup security alias, set the Component-managed authentication alias to the following value:
`jasperserver_user`
9. Click Next, review the summary information, and click Finish.

To set the connection pool size

1. In the list of JDBC data sources, click the newly created jsAuditAnalytics data source to edit it.
2. Click Additional Properties > Connection Pool Properties. You can see that Maximum Connections is set to 10 by default.
3. Set Maximum Connections to 50. You may want to set it to a higher value if necessary.
4. Click Save.

To define custom properties

1. In the list of JDBC data sources, select the checkbox for the newly created jsAuditAnalytics data source, and click Test Connection.

In the Messages area, a success or failure message appears. The failure message gives you information about which custom properties you need to define.

2. Navigate to the jsaudit data sources General Properties page.

To create optional sugarcrm and foodmart data sources

If you plan to run the sample reports, use the following values to create the foodmart and sugarcrm JNDI data sources:

1. In the list of JDBC data sources, click the link for the newly created jasperserver data source.
2. Click Save directly to the master configuration.
3. Set the connection pool size as described in [To set the connection pool size](#).

Next, deploy the WAR file in WebSphere as described in [Deploying the WAR File in WebSphere](#).

Starting and Restarting JasperReports Server

To start the jasperserver-pro application

1. Restart WebSphere.
2. In the Administrative Console, navigate to: Applications > Application Types > WebSphere Enterprise Application.
3. Select the check box next to the jasperserver-pro application and click Start.
If you make configuration changes to your JasperReports Server instance, restart it.
4. Log into JasperReports Server.

Logging into the Server

1. Go to the following URL to log in:

`http://<hostname>:9080/jasperserver-pro`

Where <hostname> is localhost, a machine name, or an IP address. The login page should appear after some time to compile the necessary JSP files.

2. Log in with administrative credentials:

User ID	Password	Description
superuser	superuser	System-wide administrator
jasperadmin	jasperadmin	Administrator for the default organization

If you have trouble logging in and get the following error message, you may be running at a WebSphere patch level that needs further configuration:

Page cannot be found, HTTP 404 error

Refer to the troubleshooting section [WebSphere Modifications](#).



The first time you log into JasperReports Server, you'll be prompted to opt-in to the JasperReports Server Heartbeat. For more information, refer to [JasperReports Server Heartbeat](#).

Refer to the JasperReports Server User Guide to begin adding reports and other resources to JasperReports Server.

Configuring Report Scheduling

Scheduled reporting in JasperReports Server allows you to run reports at specified times and gives you the option to send an email notification to users when a new report is available.

Additional Fix for Scheduled Report with JNDI Data Source

On the WebSphere application server, reports using a JNDI data source require additional configuration to correctly resolve the JNDI lookup.

The jasperserver-pro.war archive includes WebSphere-specific configuration files. These files are generated during the installation process. To enable this fix, you'll need to remove the webSphere prefix from the names of these files. Two of the file rename operations will overwrite the existing configuration file names.

Rename: WEB-INF/webSphere-applicationContext-report-scheduling-wm.xml

To: WEB-INF/applicationContext-report-scheduling-wm.xml

Rename: WEB-INF/webSphere-js.quartz.base.properties

To: WEB-INF/js.quartz.base.properties (overwrite existing file)

Rename: WEB-INF/webSphere-js.scheduling.properties

To: WEB-INF/js.scheduling.properties (overwrite existing file)



A work manager class is used to run scheduled report jobs on WebSphere. The JNDI name of the work manager and its default value (wm/default) are configured in js.scheduling.properties. The number of threads that run report jobs is provided by the work manager configuration.

Additional Change for Mail Server Authentication

If your mail server requires authentication, edit the applicationContext-report-scheduling.xml file after applying the changes above.

1. Extract the file from the WAR archive:

```
"%JAVA_HOME%\bin\jar" xf jasperserver-pro.war WEB-INF/applicationContext-report-scheduling.xml
```

2. Open the file for editing and locate the `reportSchedulerMailSender` bean.
3. Set the `javaMailProperties key="mail.smtp.auth"` value to `true`.
4. Save the file and replace it in the archive:

```
"%JAVA_HOME%\bin\jar" uf jasperserver-pro.war WEB-INF/applicationContext-report-scheduling.xml
```
5. Delete the `WEB-INF` directory that was created, along with the file it contains.

For more information about setting up report scheduling, refer to [Configuring Report Scheduling](#).

Updating XML/A Connection Definitions (Optional)

If you loaded the sample data, and you want to run the Analysis XML/A examples, you'll need to update the XML/A connection resources to use the correct web port.

The typical port used by WebSphere is 9080. Follow the procedure in [Updating XML/A Connection Definitions](#).

Troubleshooting your Configuration

Startup Problems

The most common problems are errors in the database configuration. These are typically errors in the database configuration files or in the application server configuration files. For information about resolving these errors, refer to the troubleshooting section [Troubleshooting](#).

Error Running Report

If you have trouble running reports in your new JasperReports Server instance, refer to the troubleshooting section [Error Running a Report](#). If you are having trouble running the MDX

example Topic or SugarCRM OLAP view, you need to update the port for XML/A connections. See [Updating XML/A Connection Definitions \(Optional\)](#).

Filter Error Using MySQL

The following error could be caused by an incorrect ampersand setting on your data source configuration:

```
Error 500: Filter [characterEncodingProxyFilter]: could not be
initialized
```

The data source line needs to have `&` and not `&` to be evaluated correctly. That is, the URL you enter in the procedure to define the JDBC data source and expose it through JNDI should look like this:

```
jdbc:mysql://localhost/jasperserver?useUnicode=true&amp;characterEncoding
=UTF-8
```

Error Creating Internationalized Name

If you encounter errors when creating resources with internationalized names and you have an Oracle database, configure your Oracle JDBC driver. Set the Oracle-specific option listed in the tables of [Setting JVM Options](#).

Xerces Error

In earlier releases of JasperReports Server, it was possible to find the following error in the WebSphere log:

```
SRVE0068E: Uncaught exception thrown in one of the service methods of the
servlet:
jasperserver. Exception thrown:
org.springframework.web.util.NestedServletException:
javax.xml.validation.SchemaFactoryFinder$ConfigurationError: Provider
org.apache.xerces.jaxp.validation.XMLSchemaFactory could not be
instantiated:
org.apache.xerces.impl.dv.DVFactoryException: DTD factory class
org.apache.xerces.impl.dv.dtd.DTDDVFactoryImpl does not extend from
DTDDVFactory.
```

More recently, the server uses version 2.10.0.

The error shown above is caused by a conflict between the IBM JDK used by WebSphere and the xercesImpl-2.6.2 library bundled with older versions of JasperReports Server. There are two solutions:

- Remove the xercesImpl library from the following location:

```
<websphere>\profiles\AppSrv<NN>\installedApps\<node>\jasperserver-pro_
war.ear\
    jasperserver-pro.war\WEB-INF\lib
```

- Update the xercesImpl library to a new version (if it is an old version).

OLAP View Fails with Exception

The following error may occur because AspectJ needs class loaders to be tried out in a specific order:

```
java.lang.NoSuchMethodError:
org/aspectj/runtime/reflect/Factory.makeMethodSig(
    java/lang/String;
    ...)
org/aspectj/lang/reflect/MethodSignature;
```

Change the default class loader policy:

1. In the WebSphere Administrative Console, navigate to Applications > (app-name) > Manage Modules > JasperServer UI application.
2. Change the following setting:

Property Name	Value
Class loader order	Class loaded with local class loader first (parent last)

3. Click OK.
4. Save the master configuration.
5. Restart the WebSphere server.

Installing the WAR File for WebLogic

JasperReports Server supports deployment on the WebLogic Application Server, but requires its own database to store information such as users, organizations, and the repository. WebLogic users need the WAR file distribution to install JasperReports Server. Download the WAR file distribution from [Jaspersoft Technical Support](https://www.jaspersoft.com/support) (<https://www.jaspersoft.com/support>) or contact your sales representative. The WAR file distribution comes in a file named `js-jrs_9.0.0_bin.zip`.

The WAR file distribution includes two sample databases containing data for optional demos. For evaluation, we recommend you install the sample databases. In a production environment, we advise against installing sample data of any kind. You create and initialize the required repository database and the optional sample databases before deploying JasperReports Server in WebLogic. The WebLogic administrator uses the WebLogic Administrative Console or domain `config.xml` to deploy JasperReports Server.

This chapter contains the following sections:

- [Procedure for Installing the WAR File for WebLogic](#)
- [Setting Java Properties](#)
- [Configuring Other Database Connections](#)
- [Starting the Server](#)
- [Logging into the Server](#)
- [Configuring Report Scheduling](#)
- [Restarting the Server](#)
- [Updating XML/A Connection Definitions \(Optional\)](#)
- [Troubleshooting Your JasperReports Server Configuration](#)

Procedure for Installing the WAR File for WebLogic

To meet prerequisites for installing the WAR file for WebLogic

1. Check that a supported version of the Oracle/Sun Java JDK is installed.
2. Check that the `JAVA_HOME` system environment variable points to the JDK.
3. Install the PostgreSQL, MySQL, Oracle, SQL Server, or DB2 database.



The target database can be on a remote server.

Due to limitations in the vendor's driver for MySQL, Jaspersoft has verified MySQL using the MariaDB driver.

To install the WAR file for WebLogic

1. Extract all files in `js-jrs_9.0.0_bin.zip` into a top-level directory, such as `C:\Jaspersoft` on Windows or `/home/<user>` on Linux.

Unpacking the ZIP file creates the directory `js-jrs_pro_9.0.0_bin.zip`.

2. Check that WebLogic is installed in the default location on your local machine.

If WebLogic is not installed in the default location, or if you encounter problems using the buildomatic scripts, set up the database manually as described in [Manually Creating the JasperReports Server Database](#). After setting up the database manually, skip [Step 6](#) through [Step 9](#), and proceed to [Step 10](#).

3. (If you are using MySQL, you can skip this step.) Copy your JDBC driver to WebLogic. PostgreSQL Example:

- a. Copy the JDBC jar:

From:

`<js-install>/buildomatic/conf_source/db/postgresql/jdbc`

To:

`<weblogic_home>/server/lib`

Note that the MySQL JDBC driver is included in recent versions of WebLogic.

- To ensure you have full support for import/export from the command line, copy your JDBC driver to the following location. If you are not using the command line for import/export, you can skip this step:

from: <js-install>\buildomatic\conf_source\db\<your_database>\jdbc\

to: <js-install>\buildomatic\conf_source\iePro\lib

- Restart WebLogic using the `startWebLogic.cmd/sh`.
- Copy the `.properties` file for your database:
 - From — <js-install>/buildomatic/sample_conf/
 - To — <js-install>/buildomatic
- Rename the file you copied to `default_master.properties` file.
- Edit the `default_master.properties` file to add settings specific to your database and your application server. [Sample Values for the default_master.properties File](#) shows sample property values.



When `appServerType = skipAppServerCheck`, `buildomatic` skips the application server type validation. Use this setting when installing JasperReports Server with WebLogic. Backslashes in `appServerDir` must be doubled, for example `C:\\WL\\Application_Server`. Make sure that there are no spaces in the `appServerDir` path.

Sample Values for the default_master.properties File

Database	Sample Property Values
PostgreSQL	<code>appServerType=skipAppServerCheck appServerDir=[path to WebLogic application server] dbUsername=postgres dbPassword=postgres dbHost=localhost</code>
DB2	<code>appServerType=skipAppServerCheck appServerDir=[path to WebLogic application server] dbUsername=db2inst1 dbPassword=password dbHost=localhost</code>
MySQL	<code>appServerType=skipAppServerCheck appServerDir=[path to WebLogic application server] dbUsername=root dbPassword=password dbHost=localhost</code>

Database	Sample Property Values
Oracle	<pre>appServerType=skipAppServerCheck appServerDir=[path to WebLogic application server] sysUsername=system sysPassword=password dbUsername=jasperserver dbPassword=password dbHost=hostname</pre> <p>Note that dbUsername must be the same as the Oracle user name.</p>
SQL Server	<pre>appServerType=skipAppServerCheck appServerDir=[path to WebLogic application server] dbUsername=sa dbPassword=sa dbHost=localhost</pre>

For the Split installation, configure the additional settings in the `default_master.properties` file as described in [Additional Buildomatic Configuration for Split Installation](#).

For the JNDI Restricted Access installation, configure the additional settings in the `default_master.properties` file as described in [Installation Types](#).

- Set up the database and optional sample databases using the buildomatic Ant scripts. Enter commands in the table below to call buildomatic Ant scripts:



Exception: For DB2, skip this step and perform [Step 1](#) to [Step 4](#) in [DB2](#), then go to [Step 10](#) of this procedure.

You call buildomatic Ant scripts from the command line using the following syntax:

Windows – `js-ant <target-name>`

Linux – `./js-ant <target-name>`

Buildomatic Targets to Execute

Commands	Description
<code>cd <js-install>/buildomatic</code>	Goes to the buildomatic directory.
<code>js-ant create-js-db</code>	Creates the jasperserver repository database.
<code>js-ant init-js-db-pro</code> <code>js-ant import-minimal-pro</code>	Initializes database, loads core application data.
<code>js-ant create-sugarcrm-db</code>	(Optional) Creates sample databases.

Commands	Description
<code>js-ant create-foodmart-db</code>	
<code>js-ant load-sugarcrm-db</code> <code>js-ant load-foodmart-db</code>	(Optional) Loads sample data into the sample databases.
<code>js-ant import-sample-data-pro</code>	(Optional) Loads the demos that use the sample data.
<code>js-ant create-audit-db</code>	(Optional) Creates the audit database. Required only for the Split installation.
<code>js-ant init-audit-db-pro</code>	(Optional) Initializes the audit database. Required only for the Split installation.



On non-Linux Unix platforms, the `js-ant` commands may not be compatible with all shells. If you have errors, use the `bash` shell explicitly. For more information, see [Bash Shell for Solaris, IBM AIX, HP UX and FreeBSD](#).



Installing JasperReports Server automatically generates encryption keys that reside on the file system. These keys are stored in a dedicated Jaspersoft keystore. Make sure that this keystore is properly secured and backed up, as described in the *JasperReports Server Security Guide*.

10. Add the database driver location to the WebLogic classpath in the following file:

```
<weblogic_home>/domains/<user_domain>/bin/setDomainEnv.sh
```

Use the path for the driver copied in [Step 3](#). For example, for a PostgreSQL driver, you might add the following:

```
export CLASSPATH=/opt/Oracle/Middleware/Oracle_Home/wlserver/server/lib/postgresql.jar:
/opt/Oracle/Middleware/Oracle_Home/wlserver/server/lib/jswlstc-1.0.jar:$CLASSPATH
```

11. In WebLogic, open an Administrative Console window and navigate to `Services > Data Sources` or `Domain Configurations > Services > Data Sources`.

12. Click New and then Generic Data Source for each of the data source columns in the following table, and enter the following values for a PostgreSQL database. You'll need to click Next after entering the database driver and after One-Phase Commit.



To use a database other than PostgreSQL, configure the database connections using settings shown in [Configuring Databases Drivers](#).

If you plan to use the sample databases (Foodmart and Sugar CRM), perform this step and the following step for each database.

Parameter Name	JasperReports Server	Jasper Reports Server Audit	JasperServerSystemDataBase	AuditAnalyticsDataBase	Foodmart DataBase	Sugar CRM
Name	JasperServerDataBase	AuditDataBase	Jasper Server System DataBase	Audit Analytics DataBase	Foodmart DataBase	Sugarcrm DataBase
JNDI Name	JasperServerDataBase	AuditDataBase	JasperServerSystemDataBase	AuditAnalyticsDataBase	Foodmart DataBase	Sugarcrm DataBase
Database Type	PostgreSQL					
Database Driver	PostgreSQL Driver Versions: using org.postgresql.Driver					
Supports Global Transactions	Selected					
One-Phase Commit	Selected					

13. Set connection properties. Sample properties for a PostgreSQL database are:

Parameter Name	JasperReports Server	JasperReports Server Audit	Foodmart	Sugar CRM
Database Name	jasperserver	For Compact installation: jasperserver For Split installation: jsaudit	foodmart	sugarcrm
Host Name			localhost	
Port			5432	
Database User Name			postgres	
Password			postgres	
Confirm Password			postgres	

14. Test the database connection:

- a. For SugarCRM and Foodmart, use the default connections:
 - jdbc:postgresql://localhost:5432/sugarcrm
 - jdbc:postgresql://localhost:5432/foodmart
- b. Change the URL for the JasperServerDataBase to:
 - jdbc:postgresql://localhost:5432/jasperserver
- c. Change the URL for the AuditDataBase to:
 - For Compact installation: jdbc:postgresql://localhost:5432/jasperserver
 - For Split installation: jdbc:postgresql://localhost:5432/jsaudit
- d. Change the URL for the JasperServerSystemDataBase to:
 - jdbc:postgresql://localhost:5432/jasperserver
- e. Change the URL for the AuditAnalyticsDataBase to:
 - For Compact installation: jdbc:postgresql://localhost:5432/jasperserver

For Split installation: jdbc:postgresql://localhost:5432/jsaudit

15. Select targets and ensure that AdminServer is set for all data sources.
16. In WebLogic, open an Administrative Console window and navigate to Services > Data Sources or Domain Configurations > Services > Data Sources
17. Select each created data source (JasperServerDataBase, AuditDataBase, FoodmartDataBase, and SugarcrmDataBase)
18. Select the Connection Pool tab and increase the Maximum Capacity setting, depending on load. For most installations, a Maximum Capacity in the range 50 – 100 should be sufficient. If you receive connection pool errors, increase this setting; see the documentation for WebLogic for more information.
19. On the Connection Pool tab, expand Advanced and enable Test Connections On Reserve. Set Test Frequency to 30 seconds and set Test Table Name for your datasource and database type. Sample properties for a PostgreSQL database are:

Parameter Name	JasperReports Server	Jasper Reports Server Audit	JasperServerSystemDataBase	AuditAnalyticsDataBase	Foodmart	Sugar CRM
Name	JasperServerDataBase	AuditDataBase	Jasper Server System DataBase	Audit Analytics DataBase	Foodmart DataBase	Sugarcrm DataBase
Test Table Name	JIREPORTJOB	JIREPORTJOB	JIREPORTJOB	JIREPORTJOB	ACCOUNT	ACCOUNTS

20. Click Save.

21. Use the Java jar tool or an unzip tool to unpack the jasperserver-pro.war file. For example, using the Java jar tool, enter these commands to unpack the jasperserver-pro.war file to a folder:

```
cd <js-install-dir>
mkdir jasperserver-pro
cd jasperserver-pro
"%JAVA_HOME%/bin/jar" xvf ../jasperserver-pro.war
```

22. Search for conflicting JARs and delete them from the WAR file. If the following JARs are present in your WebLogic installation, you need to delete them from your JasperReports Server installation to avoid conflicts. To do this:

a. Search your WebLogic installation for the following files:

```
jaxb-api-<ver>.jar  
jaxb-impl-<ver>.jar  
serializer-<ver>.jar  
stax-api-<ver>.jar  
xalan-<ver>.jar  
xercesImpl-<ver>.jar  
xml-apis-<ver>.jar
```

b. Change to the JasperReports Server WEB-INF/lib directory:

```
cd <js-install>/jasperserver-pro/WEB-INF/lib
```

c. Delete any conflicting JARs.

23. Replace the default web.xml file with the following commands:

```
cd <js-install-dir>/jasperserver-pro/WEB-INF  
mv ./web-version24.xml ./web.xml
```

24. Update your Hibernate, Quartz, and Mail Server configuration:

a. The buildomatic logic has already configured the `hibernate.properties` and `js.quartz.properties` files for your database type. So you can copy these files to the `jasperserver-pro` file as shown below.

Copy from:

```
<js-install>/buildomatic/build_conf/default/webapp/WEB-  
INF/classes/hibernate.properties  
<js-install>/buildomatic/build_conf/default/webapp/WEB-  
INF/js.quartz.properties
```

To:

```
jasperserver-pro/WEB-INF/classes
```


- b. Edit the scheduler URI port value for WebLogic in the `js.quartz.properties`:

Edit `js.quartz.properties`:

Set :

```
report.scheduler.web.deployment.uri=http://localhost:8080/jasperserver-pro
```

To:

```
report.scheduler.web.deployment.uri=http://localhost:7001/jasperserver-pro
```

- c. If you want to configure JasperReports Server to automatically schedule and email reports, enter your mail server information in the `js.quartz.properties` file. Modify all `report.scheduler.mail.sender.*` properties as necessary for your mail server.

Copy the `../buildomatic/keystore.init.properties` file to the `../WEB-INF/classes` directory.

Also for WebLogic (for DB2, Oracle, and SQL Server):

`progressiveStreaming=2` needs to be added to the variables.

For more information on the `progressiveStreaming` variable, refer to the section "BeanDefinitionStoreException with DB2" with Vendor's Driver in [Database-related Problems](#).

25. If your mail server requires authentication, edit the `applicationContext-report-scheduling.xml` file:
- Open the `jasperserver-pro/WEB-INF/applicationContext-report-scheduling.xml` file for editing and locate the `reportSchedulerMailSender` bean.
 - Set the `javaMailProperties` `key="mail.smtp.auth"` value to `true`.
26. Now you can change to the `jasperserver-pro` folder and re-archive the `jasperserver-pro.war` file, using commands such as the following.

Commands	Description
<code>cd ../../</code>	Changes to the <code>jasperserver-pro</code> folder
<code>mv ../jasperserver-pro.war ../BAK-jasperserver-pro.war</code>	Renames the original

Commands	Description
	jasperserver-pro.war file.
<code>"%JAVA_HOME%/bin/jar" cvf ../jasperserver-pro.war *</code>	Re-archives the jasperserver-pro.war file.
<code>cd ..</code> <code>mv jasperserver-pro BAK-jasperserver-pro</code>	Renames the unneeded working folder to a backup location.



You now have a `jasperserver-pro.war` file you can use for deploying to WebLogic.

27. Edit your WebLogic domain configuration file `<wl-domain>/config/config.xml`:



`<wl-domain>` is the path of the domain within WebLogic that contains your JasperReports Server deployment. For example `<weblogic>/samples/domains/wl_server`.

- a. Locate the `server` and `security-configuration` elements, and insert the following parameters:

```
<server>
...
    <stuck-thread-max-time>1200</stuck-thread-max-time>
    <listen-address></listen-address>
</server>
<security-configuration>
...
    <enforce-valid-basic-auth-credentials>>false</enforce-valid-basic-auth-credentials>
</security-configuration>
```

- b. Check that the `stuck-thread-max-time` element appears above the `listen-address` element before the closing `</server>` tag.



In some cases, setting the `stuck-thread-max-time` may cause a schema validation error. In this case, you can try removing this line from the configuration file.

28. Set JVM options as described in [Setting Java Properties](#).

Deploy JasperReports Server to WebLogic

1. Enable the Lock & Edit button:
 - a. Select the Preferences link at the top of the Admin console
 - b. Scroll to the bottom of the User Preferences screen and deselect Automatically Acquire Lock and Activate Changes.
 - c. Save.
2. In the Administrative Console, click the Lock & Edit button and navigate to Deployments.
3. On the Deployments page click the Install button.
4. Select the path to <js-install>. Click Next.
5. Leave the radio button selected for Install this deployment as an application. Click Next.
6. When prompted, enter the following parameter values:

Parameter Name	Example Value
Name	jasperserver-pro
Security	Custom Roles and Policies
Source accessibility	Use the defaults defined by the deployment's targets

7. Review your choices and click Finish.
8. Click Save.

Setting Java Properties

Edit the WebLogic startup script for your platform to include the settings described in the following tables. Substitute the location of your JasperReports Server license file where necessary:

WebLogic Startup Settings on Windows

Filename	<wl-domain>\bin\startWebLogic.cmd
Settings	<pre>set JAVA_OPTIONS=%JAVA_OPTIONS% -Djs.license.directory=C:\<js-install>\ -Dfile.encoding=UTF-8 -Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0 -Dlog4j.configurationFile=WEB-INF/log4j2.properties -Xms2048m -Xmx4096m -Xss2m</pre>
Java 11	<pre>set JAVA_OPTS=%JAVA_OPTS% -Djava.locale.providers=COMPAT</pre>
For Oracle (optional)	<pre>set JAVA_OPTIONS=%JAVA_OPTIONS% -Doracle.jdbc.defaultNChar=true</pre>



Setting the Oracle localization option, defaultNChar, can substantially impact the performance of JDBC queries. If you don't need to support UTF-8 for your Oracle database, you can omit this setting.

WebLogic Startup Settings on Linux

Filename	<wl-domain>/bin/startWebLogic.sh
Settings	<pre>export JAVA_OPTIONS="\$JAVA_OPTIONS -Djs.license.directory=/home/<user>/weblogic/jasperlicense/ -Dfile.encoding=UTF-8 -Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0 -Dlog4j.configurationFile=WEB-INF/log4j2.properties -Xms2048m -Xmx4096m -Xss2m"</pre>
Java 11	export JAVA_OPTS="\$JAVA_OPTS -Djava.locale.providers=COMPAT"
For Oracle (optional)	export JAVA_OPTIONS="\$JAVA_OPTIONS -Doracle.jdbc.defaultNChar=true"



Setting the Oracle localization option, defaultNChar, can substantially impact the performance of JDBC queries. If you don't need to support UTF-8 for your Oracle database, you can omit this setting.

Configuring Other Database Connections

Configuring Databases Drivers

Use these settings to connect to a database other than PostgreSQL using the database vendor's driver.

Configuring a MySQL Connection

Database Setting	Value
Host	localhost
Name or SID	For JasperServerDataBase: jasperserver For AuditDataBase: <ul style="list-style-type: none">• For Compact installation: jasperserver• For Split installation: jsaudit
User	root
Password	password
Port	3306
characterEncoding	UTF-8
autoReconnect	true
tinyInt1isBit	false
autoReconnectForPools	true
Hibernate Dialect	MySQLInnoDBDialect
Quartz Driver Delegate	StdJDBCDelegate

Enter the following for Test Table Name in [Step 19](#):

Parameter Name	JasperReports Server	Jasper Reports Server Audit	JasperServerSystemDataBase	AuditAnalyticsDataBase	Foodmart	Sugar CRM
Name	JasperServerDataBase	AuditDataBase	Jasper Server System DataBase	Audit Analytics DataBase	Foodmart DataBase	Sugarcrm DataBase
User	root	root	jasperuser	jasperuser	foodmart	sugarcrm
Test Table Name	JIReportJob	JIAuditEvent	JIReportJob	JIAuditEvent	account	accounts

Configuring an Oracle Connection

To use the Oracle driver, enter the following properties:

Database Setting	Value
Host	localhost
Name or SID	Orcl
User	For JasperServerDataBase: jasperserver For AuditDataBase: <ul style="list-style-type: none"> • For Compact installation: jasperserver • For Split installation: jsaudit
Password	password
Port	1521

Database Setting	Value
Hibernate Dialect	OracleJICustomDialect
Quartz Driver Delegate	StdJDBCDelegate

Enter the following for Test Table Name in [Step 19](#):

Parameter Name	JasperReports Server	Jasper Reports Server Audit	JasperServerSystemDataBase	AuditAnalyticsDataBase	Foodmart	Sugar CRM
Name	JasperServerDataBase	AuditDataBase	Jasper Server System DataBase	Audit Analytics DataBase	Foodmart DataBase	Sugarcrm DataBase
User	jasperserver	For Compact installation: jasperserver For Split installation: jsaudit	jasperuser	jasperuser	foodmart	sugarcrm
Test Table Name	DUAL	DUAL	DUAL	DUAL	DUAL	DUAL

Configuring a DB2 Connection

To use the DB2 driver, enter the properties below.

Database Setting	JasperReports Server	JasperReports Server Audit	Foodmart	Sugar CRM
Host	localhost			
Name or SID	JSPRSRVR	For Compact installation: JSPRSRVR For Split installation: JSAUDIT	foodmart	sugarcrm
currentSchema	JSPRSRVR	For Compact installation: JSPRSRVR For Split installation: JSAUDIT	FOODMART	SUGARCRM
User	db2inst1			
Password	password			
Port	50000			
Hibernate Dialect	DB2JICustomDialect			
Quartz Driver Delegate	DB2v8Delegate			

Enter the following for Test Table Name in [Step 19](#):

Parameter Name	JasperReports Server	JasperReports Server Audit	Foodmart	Sugar CRM
Name	JasperServerDataBase	AuditDataBase	FoodmartDataBase	SugarcrmDataBase

Parameter Name	JasperReports Server	JasperReports Server Audit	Foodmart	Sugar CRM
User	db2inst1	db2inst1	foodmart	sugarcrm
Test Table Name	JIREPORTJOB	JIREPORTJOB	ACCOUNT	ACCOUNTS

Configuring a SQL Server Connection

To use the SQL Server driver, enter the following properties:

Database Setting	SQL Server
Host	localhost
Name or SID	For JasperServerDataBase: jasperserver For AuditDataBase: <ul style="list-style-type: none"> • For Compact installation: jasperserver • For Split installation: jsaudit
User	sa
Password	sa
Port	1433
Hibernate Dialect	SQLServerJICustomDialect
Quartz Driver Delegate	StdJDBCDelegate

Enter the following for Test Table Name in [Step 19](#):

Parameter Name	JasperReports Server	Jasper Reports Server Audit	JasperServerSystemDataBase	AuditAnalyticsDataBase	Foodmart	Sugar CRM
Name	JasperServerDataBase	AuditDataBase	Jasper Server System DataBase	Audit Analytics DataBase	Foodmart DataBase	Sugarcrm DataBase
User	sa	sa	jasperuser	jasperuser	foodmart	sugarcrm
Test Table Name	jireportjob	jireportjob	jireportjob	jireportjob	account	accounts

Starting the Server

1. In the Administrative Console, navigate to Deployments.
2. Select the jasperserver-pro application and click Start.
3. In the Start Application Assistant page, click Yes.

Logging into the Server

1. To log in, go to this URL:

`http://<hostname>:7001/jasperserver-pro`

Where <hostname> could be localhost, a machine name, or an IP address. The login page appears when the necessary JSP files are compiled.

2. Enter the following credentials:

User ID	Password	Description
superuser	superuser	System-wide administrator
jasperadmin	jasperadmin	Administrator for the default organization



The first time you log in, you'll be prompted to opt-in to the JasperReports Server Heartbeat. For more information, refer to [JasperReports Server Heartbeat](#).

Refer to the JasperReports Server User Guide to begin creating reports and other resources.

Configuring Report Scheduling

Scheduled reporting allows you to run reports at specified times. And you have the option to send an email notification to users when a new report is available.

For more information about setting up report scheduling, refer to [Configuring Report Scheduling](#).

Restarting the Server

If you made configuration changes to your server instance, restart JasperReports Server.

Updating XML/A Connection Definitions (Optional)

If you have loaded the sample data and would like to run the XML/A examples, update the XML/A connection resources to use the correct web port. The typical port used by WebLogic is 7001. Follow the procedure in [Updating XML/A Connection Definitions](#).

Troubleshooting Your JasperReports Server Configuration

If you have problems running the buildomatic scripts that set up the database, set up the database manually. For more information, see [Setting JVM Options for Application Servers](#).

Startup Problems

Problems starting a new JasperReports Server instance are usually errors in the database or application server configuration files. To resolve these errors, see [Troubleshooting](#).

Error Running Report

If you have trouble running reports in your new JasperReports Server instance, see [Error Running a Report](#).

Managing Keystore Files

When you install JasperReports Server recent version on a machine in addition to versions 7.2.x and 7.5.x, the new installation or one of the previous instances may not function properly. This is caused by permissions on the keystore files needing to be different in recent releases, and differences in file contents. This issue affects only version 7.2.x and higher. Earlier versions are not affected. To resolve this issue, you need to change the .ksp path.

To change the .ksp path for deploying JasperReports Server war file



Before making the changes, you must back up the .jrsksp and .jrsk files to a secure location. And make sure all backups are readable only by admin users.

1. Decode the .jrsksp file using Base64 Table 1 of RFC 2045, url unsafe, and padded. For example:

Linux: `cat.jrsksp | openssl base64 -d > <temp_file>`

Windows: `certutil -decode .jrsksp temp.jrsksp`

2. Edit the .jrsksp file and add the ksPath property with the path to the .jrsk file in the current folder, for example:

Linux: `ksPath=~/.jrs800/.jrsk/`

Windows: `ksPath=%HOME%\jrs800\jrsks`

3. Encrypt the temp.jrsksp file back to .jrsksp

4. Re-encode the .jrsksp file with the same Base64 settings, for example:

Linux: `base64 .jrsksp > ./temp.jrsksp`

`mv temp.jrsksp .jrsksp`



mv is not required, but you should make sure that the original .jrsksp file does not exist (backed up). The command for this is: `base64 /path/to/file > .jrsksp.`

For example: `./temp.jrsksp > .jrsksp`

Windows: `certutil a- encode temp.jrsksp tmp.b64 && findstr /v /c:tmp.b64 .jrsksp`

5. Locate the buildomatic/keystore.init.properties server configuration file and edit it to modify the ks and ksp properties:

Linux: `ks=~/.jrs800/ ksp=~/.jrs800/`

Windows: `ks=%HOME%\jrs800\ ksp=%HOME%\jrs800\`

6. Use the modified buildomatic/keystore.init.properties file to overwrite the same file in the following locations (depending on the release):

buildomatic/conf_source/iePro/keystore.init.properties

buildomatic/build_conf/default/webapp/WEB-

INF/classes/keystore.init.properties

```
apache-tomcat/webapps/jasperserver-pro/WEB-INF/classes/keystore.init.properties
```

7. Restart the service.

Scalable Query Engine

The JasperReports Server scalable query engine is a new feature in version 8.0 that supports high performance reporting under load. It uses Docker containers deployed through Kubernetes to create a cluster of virtual pods that process Ad Hoc views in parallel.

The scalable query engine is completely optional: you can choose whether to deploy it during installation, at a later time, or not at all. For small deployments without performance issues, no further installation is necessary and you may skip this chapter. If you have hundreds of users and experience performance issues when displaying dashboards, this chapter explains how to install and configure the scalable query engine.

This chapter contains the following sections:

- [Overview](#)
- [Architecture](#)
- [Downloading the Software](#)
- [Docker Configuration](#)
- [Deploying with Kubernetes](#)
- [Configuring JasperReports Server](#)
- [Logging and Debugging](#)

Overview

The JasperReports Server scalable query engine runs certain Ad Hoc views in parallel on separate virtual nodes, called workers, to improve the performance. For example, a dashboard may contain several Ad Hoc views and take several seconds to display even on a server with no load. When many users open dashboards simultaneously, the load increases on the server and all users experience delays.

When the scalable query engine is deployed, the server sends dashboard Ad Hoc tasks to the workers, the workers independently query the data source and process the results, and then the completed reports are displayed seamlessly in each dashboard. With many

workers processing reports in parallel, the delays for each user are minimized. The workers have their own data cache, so repeated queries are handled efficiently. You can also configure Kubernetes to scale automatically, launching new workers when load is high and removing workers when it's low.

In addition to embedded Ad Hoc views, the engine also processes queries (but not rendering) for Ad Hoc reports and lists of input control values. In order to display a list of possible values for an input control, the server must query the data source and process a list of potentially thousands of results. The scalable query engine can do this work in parallel and benefit from its local cache, just like Ad Hoc views.

The scalable query engine runs on virtual nodes that are separate from and in addition to the host running your instance of JasperReports Server. If you choose to deploy the scalable query engine, you will need to provision these additional nodes, usually through a cloud provider.

It is important to understand which reports are handled by the scalable query engine:

- The scalable query engine applies only to embedded Ad Hoc views: those that run in dashboards and through Visualize.js.

The scalable query engine does not apply to Ad Hoc views in the designer or Ad Hoc reports in the viewer. These are always processed by the server's own Ad Hoc engine, even if they are large reports with a longer response time.

- The engine can handle the following types of data sources:
 - JDBC data source (`JdbcReportDataSource`)
 - JNDI data source (`JndiJdbcReportDataSource`), requires configuration
 - Custom data source (`CustomReportDataSource`)
 - AWS data source (`AwsReportDataSource`)
 - Azure data source (`AzureSqlReportDataSource`)

Other types of data sources such as the various big data adapters are not certified for use with the scalable query engine. Embedded Ad Hoc views with unsupported data sources are again processed by the server's own Ad Hoc engine. The user doesn't see any change in behavior, the embedded Ad Hoc view is still displayed as expected, the only difference is the scalability (performance under load). Make sure your embedded Ad Hoc views use the data sources listed above to take advantage of the scalable query engine.

- The scalable query engine processes only embedded Ad Hoc views. If you want improved performance for JRXML reports, Jaspersoft provides the JasperReports IO

(JRIO) At-Scale product that is also based on a Kubernetes cluster of autoscalable pods. Both JRIO At-Scale and the Scalable Query Engine can be deployed simultaneously with the same server, but they remain separate clusters with separate Helm charts.

The following diagram summarizes which Ad Hoc views can be processed by the scalable query engine:

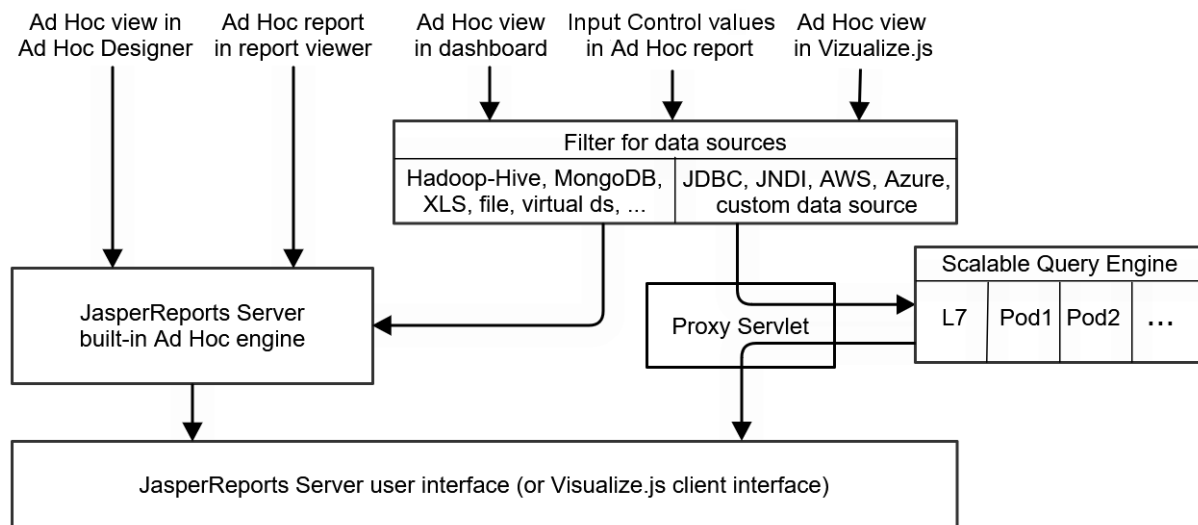


Figure 5: Ad Hoc Views Processed by the Scalable Query Engine

The scalable query engine is completely transparent: users are still logged into JasperReports Server through a browser or authenticated with Visualize.js. They view dashboards containing Ad Hoc views in their JasperReports Server sessions or Visualize.js clients as before. There is no user-visible change to the server when the scalable query engine is deployed, only a performance improvement under load.

The rest of this chapter describes the components of the scalable query engine and how to deploy them. When you are ready to deploy the scalable query engine, first install JasperReports Server WAR file distribution, then download and configure the Docker container for the Ad Hoc workers, provision your virtual machines, and finally deploy the worker pods in a cluster with Kubernetes. Jaspersoft recommends doing this during the installation process, before putting your server into production. However, it can also be done at a later time, though you will need to reconfigure and restart the server while deploying the workers.

Architecture

When deployed, the scalable query engine has a container-based architecture where multiple pods or workers process Ad Hoc views in parallel. The following figure shows its components that are further described in this section:

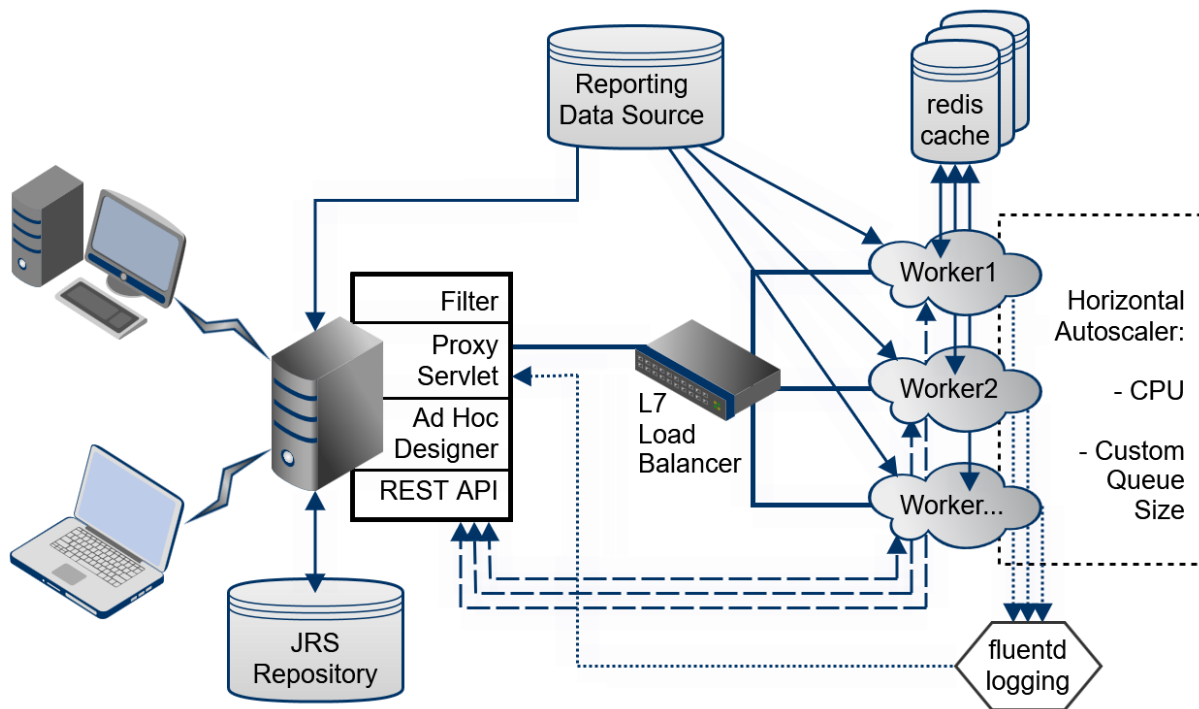


Figure 6: Architecture of the Scalable Query Engine

Filter

Once the scalable query engine is activated in JasperReports Server, a new filter processes all requests for embedded Ad Hoc views and reports. Ad Hoc views in dashboards or requested by Visualize.js clients are sent to the filter and then to the scalable query engine. Ad Hoc views in the Ad Hoc designer and Ad Hoc reports in the report viewer are handled internally by the server and never sent to the filter.

The filter then checks the data source of the embedded Ad Hoc views to make sure they can be processed by the workers. See [Overview](#) for the list of supported data sources. The filter also checks to make sure the workers are active and able to process Ad Hoc requests.

After the filter determines that an Ad Hoc view can be handled by the scalable query engine, the filter sends the embedded Ad Hoc request to the proxy servlet.

Proxy Servlet

The proxy servlet manages the communication between the server and the workers, so that the Ad Hoc view can be securely processed on a remote worker pod.

First, the proxy servlet creates the Ad Hoc task that is sent to a worker pod, including information such as the URI of the Ad Hoc view in the repository and any attributes that apply to the user session. This information is encoded in JSON web tokens (JWT) that are signed with the server's keys. These tokens allow access to the server's REST APIs that worker pods need to begin processing the Ad Hoc request, for example the metadata in the repository for the Ad Hoc view or its input controls.

If there is an error while a worker is processing an Ad Hoc task, the proxy servlet handles retries and displays any error messages.

Load Balancer

As part of the Kubernetes cluster that manages the workers, the layer 7 load balancer distributes Ad Hoc requests to the available workers. By default, requests are queued, and it uses a round-robin algorithm to select a worker for each request.

On Kubernetes, Ingress is the load balancer by default.

Worker Pods

After receiving an Ad Hoc request, a worker pod does the querying and processing to render an Ad Hoc view. The pods are deployed in Kubernetes, usually on virtual machines in a cloud, and the general sequence of events is as follows:

1. When the worker receives the request, it is given the URI of the Ad Hoc view in the repository. It then accesses the repository through the server's REST API to get the data source and query needed for the report.
2. Using other context such as attributes and input controls, the worker determines the final query needed to obtain the dataset for the report.

3. Before sending the query to the data source, the worker first accesses the redis cache to determine if the dataset is available without running the query again. Access to the cache is based on the user who requested the Ad Hoc view and the query, so that all data access is secure.
 - a. If the query is already in the redis cache, the dataset of the results is sent from the cache to the worker.
 - b. If the query was not found in the redis cache, the query is sent to the reporting data source, and the worker waits for the new dataset. This new query result is then also added to the redis cache with its query string.
4. The worker performs the in-memory processing of the dataset, for example, doing the grouping or aggregation that is necessary in the required table, crosstab, or chart.
5. The data processing results are also stored in the redis cache for later re-use.
6. The worker generates the required table, crosstab, or chart, and through the proxy servlet, publishes it in the container that is associated with this Ad Hoc task, for example embedded in the server's dashboard interface or in a Visualize.js client.

Redis Cache

The redis cache is a high-performance distributed data store shared by all worker pods and is itself managed in the Kubernetes cluster as separate pods.

The redis cache holds the results of Ad Hoc queries, which can be huge datasets, and workers can get these results without performing the correspondingly long queries while they are held in the cache. The redis cache is actually composed of several separate caches:

- The main cache for datasets resulting from queries.
- A cache for Ad Hoc view output that has already been processed and is ready to display.
- A cache of the Ad Hoc view descriptors from the JasperReports Server repository.
- A cache for attributes associated with a given Ad Hoc view and user.

The Ad Hoc workers check each of these caches before making a request for the corresponding contents. For example, before calling the repository REST API to get the

report metadata, the worker checks the descriptors cache to see if that descriptor has already been requested and is still valid.

All the caches have timeout values that you can configure to determine how long datasets and descriptors are valid before needing to be reloaded.

Autoscaler

The Horizontal Pod Autoscaler manages the Kubernetes cluster and can launch new worker pods or remove them as needed. Through the configuration, you can set rules for scaling based on CPU usage and, optionally, memory queue size.

When the scalable query engine is live in production, the autoscaler then starts new worker pods automatically to improve performance, or stop unused pods to save on virtual machine costs.

Fluentd Logging

Because the worker pods are running on virtual nodes that can be difficult to access, the scalable query engine enables a fluentd chart to collect the logs from all the nodes. The workers use the log4j2 library, and you can set logging levels when configuring the helm chart. Then use Elasticsearch to aggregate and search the logs.

When there are errors while processing an Ad Hoc view, you can then use command-line tools on the server to view the aggregated logs from the workers and pinpoint the problem.

Downloading the Software

The scalable query engine is contained in `scalable-query-engine-9.0.0.jar` in the WAR (web archive) file distribution of JasperReports Server. There are also additional files for Docker and Kubernetes in a github that you can clone. You do not need an account on github.com, but you do need the git software locally on the host where you download and install JasperReports Server.

Prerequisites

This deployment is based on a licensed installation of the JasperReports Server WAR file distribution, enterprise edition 9.0.0 or later. The scalable query engine is not supported with the JasperReports Server binary installer, any evaluation version, nor the Community Project. You must complete the installation and configuration of the server, in particular the keystore (either generated new or imported from a previous version) before deploying the scalable query engine. See the rest of this JasperReports Server Installation Guide or the JasperReports Server Upgrade Guide to complete your server installation before continuing.

You will also need the following third-party software for this deployment:

- git
- Docker engine (19.x+) including Docker Compose V2 (3.9+)
- Kubernetes (1.19+) including kubectl
- Helm 3.5

This manual assumes this software is installed and available in your path. You should be proficient with the concepts and commands for these tools.

Downloading From Git

After installing the JasperReports Server WAR file distribution, you need to download the github project that contains folders for Docker and Kubernetes.

1. Go to the base folder of the unzipped WAR file distribution, referred to as `<js-install>` in this manual. The content of the github repository must be placed in this folder:

```
cd jasperreports-server-pro-9.0.0-bin
```

2. Run the following command to download the project from github:

```
git clone git@github.com:tibco/jaspersoft-containers.git
```

The `jaspersoft-containers` project in git includes the following top-level folders:

Folder	Description
Docker/scalableQueryEngine	Docker components for the Scalable Query Engine, described in Docker Configuration .
Docker/jrs	Docker components for a containerized version of JasperReports Server, which is beyond the scope of this document.
K8s/scalableQueryEngine	Kubernetes components for the Scalable Query Engine, described in Deploying with Kubernetes .
K8s/jrs	Kubernetes components for the containerized version of JasperReports Server, which is beyond the scope of this document.

Docker Configuration

The first step of installing the Scalable Query Engine is to configure and build the Docker images of the worker pods.

In the git project, `jaspersoft-containers/Docker/scalableQueryEngine/` contains the following files and folders:

File or Folder	Description
Dockerfile	The main installation script for the Scalable Query Engine.
.env	Environment variables for building the Docker images, see the next section.
Dockerfile.drivers	Script that copies the supported JDBC Drivers from JasperReports Server to the Scalable Query Engine.
docker-compose.yaml	Compose file to orchestrate the Scalable Query Engine, its drivers, and the redis services.

File or Folder	Description
scripts	Folder containing scripts for the Scalable Query Engine.
resources/drivers	Folder created by scripts with a copy of the JDBC drivers.
resources/keystore	Folder where you place a copy of the JasperReports Server keystore.
resources/properties	Folder containing properties files for the Scalable Query Engine.

Configuring the Docker Images

Before building the Docker images, edit the .env file to specify the following properties:

Property	Description
JASPERREPORTS_SERVER_VERSION	JasperReports Server release version (must be 9.0.0 or higher).
SCALABLE_QUERY_ENGINE_IMAGE_NAME	Name for the Docker image, <code>scalable-query-engine</code> by default.
SCALABLE_QUERY_ENGINE_DRIVER_IMAGE_NAME	Name for the JDBC drivers Docker image, <code>scalable-adhoc-drivers</code> by default.
SCALABLE_QUERY_ENGINE_DRIVER_IMAGE_TAG	Docker tag for the Scalable Query Engine image, 9.0.0 by default.
SCALABLE_QUERY_ENGINE_IMAGE_TAG	Docker tag for the Scalable Query Engine Drivers image, 9.0.0 by default.
JDK_BASE_IMAGE	Docker image certified for the version of JasperReports Server being deployed, either <code>openjdk:11-jdk</code> for Debian or <code>amazoncorretto:11</code> for Amazon Linux 2.
ks	Path to the server's .jrsks keystore file, usually

Property	Description
	/etc/secrets/keystore.
ksp	Path to the server's .jrsksp keystore properties file, usually /etc/secrets/keystore.
RELEASE_DATE	Release date of the JasperReports Server version, for example Month-Day, Year when using 9.0.0.

Building the Docker Images

The recommended way to build the Docker images for the Scalable Query Engine is to use `docker-compose` command as follows:

```
cd <js-install>/jaspersoft-containers/Docker/scalableQueryEngine
docker-compose build
```

If there is an error such as "requested access to the resource is denied," then run the following commands:

```
sudo gpasswd -a $USER docker
newgrp docker
```

Alternatively, you can also build the Docker images with the following commands:

```
cd <js-install>
docker build -t scalable-query-engine:<docker-tag>
    -f jaspersoft-containers/Docker/scalableQueryEngine/Dockerfile
.
docker build -t scalable-query-engine-drivers:<drivers-tag>
    -f jaspersoft-
containers/Docker/scalableQueryEngine/Dockerfile.drivers .
```

Deploying with Docker Alone (Optional)

If you want to install the Ad Hoc worker with Docker alone, the deployment is simpler, but you do not have the advantages of a cluster managed through Kubernetes. If you want to deploy with Kubernetes and Helm charts, skip this section and go to [Deploying with Kubernetes](#).

Edit the file `docker/application.properties` to make any necessary configuration. In particular, any JNDI data source to be used in Ad Hoc views and reports needs to be configured in this properties file.

1. Copy your server's keystore and keystore properties files to `<js-install>/jaspersoft-containers/Docker/scalableQueryEngine/resources/keystore`. The keystore files must be same that JasperReports® Server used when creating its repository database.
2. Navigate to the `<js-install>` directory and run the following command:

```
docker network create jrs_default
docker-compose up -d scalable-query-engine
```

You can check that the worker is running at:

```
<worker_ip>:8081/actuator/health.
```

3. Configure your JasperReports Server instance as described in [Configuring JasperReports Server](#), then restart your server.

Once your server and query engine are running, you can test a dashboard that contains an Ad Hoc view. After it runs, you can check the worker logs with the following command on the server:

```
docker logs scalable-query-engine/8.0.0
```

Deploying with Kubernetes

After configuring and building the Docker images, the next step is to configure Kubernetes and Helm charts to deploy the worker pods.

In the git project, `jaspersoft-containers/K8s/scalableQueryEngine/helm/` contains the following files and folders:

File or Folder	Description
<code>Chart.yaml</code>	Helm chart for the scalable query engine.
<code>values.yaml</code>	Configuration values for the Helm chart.
<code>charts</code>	Folder containing dependencies.
<code>config/jndi.properties</code>	Configuration file for JNDI data sources.
<code>secret/keystore</code>	Folder where you place a copy of the JasperReports Server keystore.

Creating a Secret

The configuration files in this section contain passwords for your server and databases. If you store passwords in files, you should manage your permissions carefully to prevent unwanted access. An alternative is to store passwords in a secret, a separate data structure managed by Kubernetes. For details about secrets, see the [Kubernetes documentation](#).

Use the following command to create a secret containing your passwords. Note that commands are often stored in a history, therefore it is best to create a script to run these commands:

```
kubectl create secret generic jrs-credentials --from-
literal=appCredentialsSecretName=password
--from-literal=foodmart.password=password --from-
literal=audit.password=password
```

You can then reference this secret inside the configuration files, for example:

```
appCredentialsSecretName=jrs-credentials
```

Configuring the Helm Chart

The tables in this section describe the properties you can set in the values.yaml file.

General settings:

Property	Description
replicaCount	The number of workers to create, but has no effect if autoscaling is enabled below. The default is 1.
jrsVersion	The version of your JasperReports Server release, by default 8.0.0.
image.tag	Tag of the scalable query engine Docker image, by default 8.0.0.
image.name	Name of the scalable query engine Docker image, which should be scalable-query-engine.
image.pullPolicy	The Docker image pull policy, by default IfNotPresent.
image.PullSecrets	If you have customized your Docker image to pull from a private registry , specify the secret here, otherwise leave null.
image.nameOverride	Overrides the default image name; can be left as "".
image.fullNameOverride	Overrides the full image name; can be left as "".

The data source properties have default values for the foodmart and sugarcrm sample data sources that you can use with the sample dashboards. In production, you should redefine config/jndi.properties for your own data sources as shown in [Specifying a JNDI Data Source](#), and then set the corresponding values here.

Property	Description
foodmart.jdbcUrl	The default URL of the foodmart data source.
foodmart.username	The username to access the foodmart data source, by default

Property	Description
	postgres.
foodmart.password	The password for the foodmart data source user. The default is postgres, but it must be entered in base64 encoded format.
sugarcrm.jdbcUrl sugarcrm.username sugarcrm.password	URL, password, and username for the sugarcrm sample data source, in the same format as foodmart. The default values are also postgres.
audit.enabled	Whether audit monitoring is enabled on JasperReports Server, by default false. When set to true, workers will attempt to write audit events to the database specified below.
audit.jdbcUrl	The URL of the database for writing audit events, by default this is the same as the server's repository. If you have a split installation with a separate audit database, specify its URL instead.
audit.userName	The username to access the audit database.
audit.password	The password to access the audit database, in base64 encoded format.
appCredentialsSecretName	Instead of storing passwords in this file, you can manually create a secret to store the server password. See Creating a Secret .

The following table contains environment properties for the workers. For more information, see the Kubernetes documentation links in the descriptions. There are additional properties in the file `templates/app-configmap.yml`, mainly for the Spring configuration on the workers.

Property	Description
timeZone	The default timezone that the workers use when

Property	Description
	processing reports, for example "America/Los_Angeles". The time zone names are those supported by <code>java.time.ZoneID</code> , which are defined in the tz database .
<code>securityContext.capabilities.drop</code>	Drops the Linux host capabilities; the default value is ALL. See the Kubernetes documentation about the security context .
<code>securityContext.runAsNonRoot</code>	Runs the worker application as non-root user when true (the default).
<code>securityContext.runAsUser</code>	Specifies the userid to run the worker application; the default is 11099.
<code>securityContext.allowPrivilegeEscalation</code>	Whether to allow the container to have host privileges; the default is false.
<code>Service.type</code>	The service type for workers should be <code>ClusterIP</code> .
<code>Service.port</code>	The service port should be set to 8080.
<code>serviceAccount.enabled</code>	Enables the service account on the workers; true by default.
<code>serviceAccount.annotations</code>	Annotations for the service account; empty <code>{}</code> by default.
<code>serviceAccount.name</code>	Name of the service account, by default <code>query-engine</code> .
<code>rbac.create</code>	Whether to create a role to use role-based access control ; true by default.
<code>rbac.name</code>	Name of the role; the default should be <code>query-engine-role</code> .
<code>extraEnv.javaopts</code>	String to add <code>JAVA_OPTS</code> to the worker's environment variables .

Property	Description
<code>extraEnv.normal</code>	Additional key=value pairs to add to environment variables; there are none by default (null value).
<code>extraEnv.secrets</code>	Specify environment variables in secrets or configmaps ; there are none by default (null value).
<code>extraVolumeMounts</code>	Adds volume mounts for storage volumes ; empty {} by default.
<code>extraVolumes</code>	Adds storage volumes; empty {} by default.

Health check properties:

Property	Description
<code>healthcheck.enabled</code>	Enables the health check so Kubernetes can detect when workers are busy or down; the default is true. See the Kubernetes documentation on liveness and readiness probes .
<code>healthcheck.livenessProbe.*</code>	<p>The liveness probe checks whether the worker is blocked or has crashed. It has the following properties:</p> <ul style="list-style-type: none"> <code>port</code> - Port of the worker, by default 8080. <code>initialDelaySeconds</code> - Time after startup when the probe begins checks, by default 120 (2 minutes). <code>failureThreshold</code> - How many failures before the worker is restarted (or marked unready), by default 24 times the period of the check. <code>periodSeconds</code> - How often the check is performed, by default 10 seconds. <code>timeoutSeconds</code> - How long the probe waits for a response before failing the check, by default 4 seconds.
<code>healthcheck.readinessProbe.*</code>	The readiness probe checks when the worker is running but unable to process requests. It has the same properties and defaults as the liveness probe, except the value of <code>initialDelaySeconds</code> is 60 (one minute).

CPU and memory resources:

Property	Description
<code>resources.enabled</code>	Whether or not the resource requests and limits are applied, by default true.
<code>resources.limits.cpu</code>	The most CPUs that each worker is allowed to use, by default 3.
<code>resources.limits.memory</code>	The most memory that each worker is allowed to use, by default, 4Gi (gibibytes).
<code>resources.requests.cpu</code>	The least number of CPUs that will be available to the worker, by default 2.
<code>resources.requests.memory</code>	The least amount of memory that will be available to the worker, by default 2 Gi (gibibytes).
<code>engineProperties.sharedCacheExpiration</code>	<p>The length of time that cache contents are valid, by default 20m (minutes). Set this value depending on your dataset size, data update intervals, and repeated report viewing. Longer cache expiration speeds up report display times, but it takes up cache space and may not display instantaneous data.</p> <p>There is also an issue with saved report options that do not apply if the report has run with previous values and is still in the cache. If you are having issues with report options not being applied in a report, lower the cache expiration, for example to 5m. The new values will apply when the report is generated after the previous report expires in the cache.</p>

Ingress load balancer:

Property	Description
<code>ingress.enabled</code>	Whether the ingress load balancer is enabled, allowing the

Property	Description
	cluster to have multiple pods and implement stickiness, by default true.
<code>ingress.hosts.host</code>	Adds the valid DNS hostname to access the scalable query engine, by default null (no value).
<code>ingress.hosts.paths.path</code>	Application context path, by default <code>/query-engine</code> .
<code>ingress.hosts.paths.pathType</code>	The path type, by default <code>Prefix</code> .
<code>ingress.tls[0].secretName</code>	Adds TLS secret name to allow secure traffic, by default null (no value).

Redis properties:

Property	Description
<code>rediscluster.enabled</code>	Enables the redis cluster for caching, by default true.
<code>rediscluster.externalRedisClusteraddress</code>	If you want to use an existing redis cluster, specify its address here, for example <code>redis://redis-cluster:6379</code> . By default, this is empty <code>{}</code> and the <code>redis-cluster.*</code> properties are used to create the redis pods.
<code>rediscluster.externalRedisClusterpassword</code>	If you specify an external redis cluster, specify its password here, otherwise empty <code>{}</code> by default.
<code>redis-cluster.nameOverride</code>	If no external redis cluster is given, Kubernetes will create a new one and give it this name for identification, by default <code>query-engine-redis-cluster</code> .
<code>redis-cluster.cluster.nodes</code>	Number of nodes to create in the redis cluster, by default 6.
<code>redis-cluster.persistence.size</code>	Size of each redis node, by default 8Gi (gibibytes).

Property	Description
<code>global.redis.password</code>	Create a password for the redis cluster.

Autoscaling properties:

Property	Description
<code>autoscaling.enabled</code>	Enables the HorizontalPodAutoscaler (HPA) for the ; the default is true. The metrics should also be enabled so they are available for the autoscaler to work.
<code>autoscaling.*</code>	The following properties define the behavior of the autoscaler: <ul style="list-style-type: none"> <code>minReplicas</code> - Minimum number of active workers, by default 2. <code>maxReplicas</code> - Maximum number or active workers, by default 10. <code>targetCPUUtilizationPercentage</code> - Minimum average CPU load of active workers to create a new worker (scale up), by default 50%. <code>targetMemoryUtilizationPercentage</code> - Minimum average memory usage on active workers to create a new worker (scale up), by default not specified {}. <code>scaleDown.stabilizationWindowSeconds</code> - Time to wait with no activity to remove a worker (scale down), by default 300 (5 minutes).
<code>customMetricScaling.enabled</code>	If you want to implement the custom metrics-based autoscaling, set this to true; the default is false. This enables the Prometheus-based autoscaling that uses the number of queued Ad Hoc tasks for scaling up. It can also be further customized for other metrics, although that is beyond the scope of this document.
<code>scalable-query-engine-scaling.*</code>	Properties for the Prometheus-based autoscaler. The name and function of each property is the same as for <code>autoscaling.*</code> , except for the following:

Property	Description
	<ul style="list-style-type: none"> averageQueuedExecutions - Minimum average queue length on active workers to create a new worker, by default 10.

Ingress controller properties that configure the load balancing. It is also possible to configure a different load balancer such as AWS by modifying templates/internal-ingress.yaml, but the details are beyond the scope of this document:

Property	Description
ingressClass	By default intranet.
kubernetes-ingress.nameOverride	By default query-engine-ingress.
kubernetes-ingress.controller.replicaCount	Number of ingress controller replicas, by default 1.
kubernetes-ingress.controller.service.type	By default LoadBalancer.
kubernetes-ingress.controller.ingressClass	By default intranet.
kubernetes-ingress.controller.config.timeout-connect	By default 30s (seconds).
kubernetes-ingress.controller.config.timeout-check	By default 60s (seconds).
kubernetes-ingress.controller.config.timeout-client	By default 240s (seconds).
kubernetes-ingress.controller.config.timeout-server	By default 240s (seconds).

Property	Description
kubernetes-ingress.defaultBackend.replicaCount	By default 1.
server.tomcat.connectionTimeout	Timeout request in milliseconds for a Tomcat request, by default 300000 (5 minutes).

Server properties for the workers to access the JasperReports Server instance through the REST API:

Property	Description
jrs.server.scheme	Protocol to access JasperReports Server, used to create the server URL, by default http.
jrs.server.host	String to create the hostname of your JasperReports Server instance to the workers within the cluster (behind the ingress load balancer).
jrs.server.port	Port number of your JasperReports Server instance, by default 80.
jrs.server.path	Path in the URL to access the server through the REST API, by default jasperserver-pro/rest_v2.
jrs.server.username	Username to access the server's REST API. By default this is jasperadmin, but you might need to change it if you have multiple organizations.
jrs.proxy.enabled	Enables the proxy for the scalable query engine, by default true.
jrs.proxy.scheme	Protocol for access through the proxy, by default http.
jrs.proxy.host	String to create the hostname of the proxy to the workers within the cluster (behind the ingress load balancer).
jrs.proxy.port	Port number of the proxy, by default 80.

Property	Description
<code>jrs.proxy.path</code>	Path in the URL of the proxy, by default <code>rest_v2</code> .
<code>jrs.proxy.username</code>	Username to reply to the proxy. By default this is <code>jasperadmin</code> , but you might need to change it if you have multiple organizations.
<code>jrs.proxy.timeout</code>	Timeout in milliseconds when replying to the proxy, by default <code>30000</code> .

JDBC driver properties for copying them to the workers:

Property	Description
<code>drivers.enabled</code>	Whether or not Kubernetes will copy the JDBC driver JARs to the workers, by default <code>true</code> .
<code>drivers.image.tag</code>	Tag of the drivers image to copy from, by default <code>8.0.0</code> .
<code>drivers.image.name</code>	Name of drivers image, by default <code>null</code> (empty value).
<code>drivers.image.pullPolicy</code>	Pull policy for the the drivers image, by default <code>IfNotPresent</code> .
<code>drivers.jdbcDriversPath</code>	Destination path where JDBC drivers are copied in the workers, by default <code>/usr/lib/drivers</code> .

Logging-related properties; for more information, see [Logging and Debugging](#):

Property	Description
<code>metrics.enabled</code>	Enables the metrics for Prometheus-based autoscaling, by default <code>false</code> .
<code>kube-prometheus-stack.prometheus-node-exporter.hostRootFsMount</code>	Whether or not to mount Prometheus in the host file system, by default <code>false</code> .

Property	Description
<code>kube-prometheus-stack.grafana.service.type</code>	By default NodePort.
<code>logging.enabled</code>	Enables the Elasticsearch, Fluentd and Kibana (EFK) logging, by default false.
<code>logging.level</code>	Logging level in the workers, by default INFO.
<code>logging.pretty</code>	Logging format in the workers, by default false.
<code>fluentd.imageName</code>	By default fluent/fluentd-kubernetes-daemonset.
<code>fluentd.imageTag</code>	By default v1.12.3-debian-elasticsearch7-1.0.
<code>fluentd.esClusterName</code>	Elasticsearch cluster name, by default elasticsearch.
<code>fluentd.esPort</code>	Elasticsearch port number, by default 9200.
<code>elasticsearch.replicas</code>	Number of pods for Elasticsearch, by default 1.
<code>elasticsearch.volumeClaimTemplate.resources.requests.storage</code>	By default 10Gi (gibibytes).
<code>kibana.service.type</code>	By default NodePort.

Specifying a JNDI Data Source

In order to use a JNDI data source, you must specify its JDBC parameters in the file `config/jndi.properties`. Remove the sample databases, and add one or more JNDI data sources as follows:

Property	Description
<code>jndi.dataSources[i].*</code>	Array of data source properties where <code>i</code> is zero-based.

Property	Description
<code>.name</code>	The name of JDBC database to use with JNDI, in the format <code>jdbc/<jdbc-name></code> . <code>.auth=</code> The type of authentication, by default Container.
<code>.factory</code>	The Java factory class to use, by default <code>com.jaspersoft.jasperserver.tomcat.jndi.JSCommonsBasicDataSourceFactory</code> .
<code>.driverClassName</code>	JDBC driver class for this database. The JAR file for this driver must be copied into the Docker image.
<code>.url</code>	JDBC URL to access the database, for example <code>jdbc:postgresql://<hostname>:<port>/<database></code> .
<code>.username</code>	Database username.
<code>.password</code>	Database user password, or configure the password in a secret and provide its name here.
<code>.accessToUnderlyingConnectionAllowed</code>	By default true.
<code>.validationQuery</code>	Simple query to test the connection, usually <code>SELECT 1</code> .
<code>.testOnBorrow</code>	By default true.
<code>.maxActive</code>	The maximum number of connections to allocate in the connection pool, for example 100.
<code>.maxIdle</code>	The maximum number of connections to maintain in the pool when they are idle, for example 30.
<code>.maxWait</code>	When all connections are in use, the duration in milliseconds that the pool will let a request wait before returning a timeout. The default is 10000 (10 seconds).

If you want to update the JNDI properties after having deployed the workers with Kubernetes, you will need to restart or redeploy the workers.

Deploying to Kubernetes

Use the following procedure to deploy the cluster of workers using Kubernetes:

1. Download the Docker images and Helm charts from github.com as described in [Downloading the Software](#).
2. Configure and build the Docker images as described in [Docker Configuration](#).
3. Add the helm dependencies with the following commands:

```
helm repo add prometheus-community https://prometheus-  
community.github.io/helm-charts  
helm repo add haproxytech https://haproxytech.github.io/helm-charts  
helm repo add bitnami https://charts.bitnami.com/bitnami  
helm repo add elastic https://helm.elastic.co
```

4. Update the helm dependencies when needed with the following commands:

```
cd <js-install>/jaspersoft-containers/K8s  
helm dependencies update scalableQueryEngine/helm
```

5. If you haven't done so already, configure the Helm chart in `values.yaml` for the deployment of Redis, Ingress, and the workers as described in [Configuring the Helm Chart](#). You can also set individual values by adding `--set <parameter_name>=<parameter_value>` to the Helm commands below.
6. If you haven't done so already, configure your JNDI data sources as described in [Specifying a JNDI Data Source](#).
7. Now you can deploy the workers on Kubernetes with the following command:

```
helm install engine scalableQueryEngine/helm
```

8. Get the ingress external IP address or nodeport and check the workers' status at:
`<ingress-IP>/query-engine/actuator/health`
9. Configure your JasperReports Server instance as described in [Configuring JasperReports Server](#), then restart your server.

Once your server and query engine are running, you can test a dashboard that contains an Ad Hoc view. After it runs, you can check the logs as described in [Logging and Debugging](#).

Configuring JasperReports Server

After deploying the Scalable Query Engine through Docker and Kubernetes and verifying that the workers are active, you can configure your JasperReports Server instance to use the workers.

Locate the `js.config.properties` file in the server instance and set the `scalableQueryEngine.enabled` and `scalableQueryEngine.url` properties. You can also set other properties such as the number of connections and length of timeouts as shown in the file below. On the server instance, you can also set the environment variable `SCALABLE_QUERY_ENGINE_URL` to the hostname of the ingress load balancer.

When done, restart your JasperReports Server instance.

```
# enables filter
scalableQueryEngine.enabled = true

# url where proxy servlet will redirect all calls (exposed from k8s), same
# as jrrio.url
# for Docker-only deployment, use <worker_ip>:8081
scalableQueryEngine.url = http://<ingress_ip>:8088

# supported datasources
scalableQueryEngine.dataSourceTypes=

com.jaspersoft.jasperserver.api.metadata.jasperreports.domain.JdbcReportDa
taSource,
com.jaspersoft.jasperserver.api.metadata.jasperreports.domain.JndiJdbcRepo
```

```
rtDataSource,  
com.jaspersoft.jasperserver.api.metadata.jasperreports.domain.CustomReport  
DataSource,  
com.jaspersoft.jasperserver.api.metadata.jasperreports.domain.AwsReportDat  
aSource,  
com.jaspersoft.jasperserver.api.metadata.jasperreports.domain.AzureSqlRepo  
rtDataSource  
  
# periodic check of provided scalableQueryEngine.url to see if system is OK  
scalableQueryEngine.livenessCheck.url = /actuator/health  
scalableQueryEngine.livenessCheck.periodSeconds = 30  
  
# in case of a failure execute on jrs, but these are only HTTP 50x errors  
and task rejection  
scalableQueryEngine.recover.enabled = false  
  
#Defines the socket timeout (SO_TIMEOUT) in milliseconds, which is the  
timeout for waiting for data or, put differently, a maximum period  
inactivity between two consecutive data packets).  
#A timeout value of zero is interpreted as an infinite timeout. A negative  
value is interpreted as undefined (system default).  
scalableQueryEngine.http.read.timeout=240000  
  
#Returns the timeout in milliseconds used when requesting a connection from  
the connection manager. A timeout value of zero is interpreted as an  
infinite timeout.  
#A timeout value of zero is interpreted as an infinite timeout. A negative  
value is interpreted as undefined (system default).#/** A integer parameter  
name to set the connection request timeout (millis) */  
scalableQueryEngine.http.connectionrequest.timeout=120000  
  
# A integer parameter name to set max connection number. Defines the  
overall connection limit for a connection pool. 10 by default is if not set  
scalableQueryEngine.http.maxConnections=1000  
scalableQueryEngine.http.maxConnectionsPerRoute=100  
  
# Determines the timeout in milliseconds until a connection is established.  
A timeout value of zero is interpreted as an infinite timeout.  
# A timeout value of zero is interpreted as an infinite timeout.  
# A negative value is interpreted as undefined (system default).  
scalableQueryEngine.http.connect.timeout=120000  
  
# The CircuitBreaker uses a sliding window to store and aggregate the  
outcome of calls. In case of to many errors either stop sending or go to  
recover, more options to cover  
scalableQueryEngine.circuitBreaker.disabled = true
```

Logging and Debugging

There are several ways to collect and analyze logs from the Ad Hoc workers running in Kubernetes. These logs are useful in case certain Ad Hoc views are causing errors. Setting log levels and searching for log messages can help debug the cause.

Log levels using Log4j2 can be set in values.yaml or using the following command line:

```
helm --set logging.level=DEBUG
```

Aggregate to File

The scalable query engine can use fluentd to collect logs from all running workers and make them available on machine where JasperReports Server is running. By default, the scalable query engine uses one fluentd instance per physical node (or virtual machine) for collecting log messages in text format from the workers in Kubernetes. There are then two possible ways to access the logs:

- Fluentbit on each physical node or VM forwards text logs to the fluentd aggregator, and aggregate logs can be viewed on the same machine as JasperReports Server.
- Fluentbit is set to JSON format on each physical node or VM and accessed directly by Elasticsearch and Kibana (ELK) to view logs. Alternatively, you can use ELK to view the aggregate text log file in the first case.

First you should download the fluentd package with the following commands:

```
git clone \
  --depth 1 \
  --filter=blob:none \
  --no-checkout \
  https://github.com/fluent/fluentd-kubernetes-
  daemonset/tree/master/docker-image/v1.11/debian-s3 \
;
mv debian-s3 docker-fluentd-file
cd docker-fluentd-file
git checkout master -- d1
```

To enable the fluentd aggregator set `logging.enabled=true` and configure the `fluentd.*` properties in `values.yaml`.

Elasticsearch Kibana (ELK)

Elasticsearch is an analytics engine from distributed sources, Kibana is a graphing package for Elasticsearch, and together they can be used to access logs from the Ad Hoc workers.

Jaspersoft does not support for Elasticsearch and Kibana, only a basic configuration for demonstration and a collector (fluentd).

1. Configure the `elasticsearch.*` properties in `values.yaml`.
2. Install Kibana with the following command:

```
helm install kibana elastic/kibana --set service.type=NodePort --set resources.requests.memory=1Gi
```

Note that the resources values (1Gi) is an example that should be modified for your own usage. The Kibana index pattern is `scalable-adhoc-<ReleaseName>-*`, where `<ReleaseName>` is the helm release name.

3. The default output format is text. If you want to log directly to ELK, you can set the output format to JSON with the following helm setting:

```
--set logging.layout=json
```

Depending on the log driver being used, JSON output may require you to configure merging and indexing on Kubernetes. It is also possible to use ELK for the visualization of text output from the fluentd aggregator.

The JSON output may require you to configure merging and indexing on K8s (depending on log driver used). The main use case would be to configure the Scalable Query Engine to log directly to ELK. For more information, see <https://docs.fluentbit.io/manual/pipeline/outputs>.

Troubleshooting

This appendix contains the following sections:

- [Binary Installer Freezes](#)
- [Error Running Buildomatic Scripts](#)
- [Unable to Edit Files on Windows 10](#)
- [Bash Shell for Solaris, IBM AIX, HP UX and FreeBSD](#)
- [Linux Issues](#)
- [Installation Error with Windows Path](#)
- [Mac OSX Issues](#)
- [Database-related Problems](#)
- [Application Server-related Problems](#)
- [License-related Errors](#)
- [Problems Importing and Exporting Data from the Repository](#)
- [Problems with Upgrade](#)

Binary Installer Freezes

If you run the JasperReports Server installer on any platform and the installation fails, the following resources can help you find the source of the error.

Installer Log Files

If you get an error when running the JasperReports Server installer on any platform, look at the log file created by the installer. This log records the status and completion of installer operations. If a specific error occurred, you may find an explicit error message. Even without an explicit error message, the log file should help you locate the cause of the error.

You'll find the installer log for your platform in the following location:

Windows:

```
<js-install>/installation.log
```

Linux:

```
<js-install>/installation.log
```

Mac

```
<js-install>/installation.log
```

If you've tried multiple installs, make sure you view the most recent install log. Then you can submit the installation.log to [Jaspersoft Technical Support](https://www.jaspersoft.com/support) (<https://www.jaspersoft.com/support>) .

Installer DebugTrace Mode

You can also run the installer a second time using the `--debugtrace` option. This creates a binary output file with precise details about the execution of the installer and any problems encountered. [Jaspersoft Technical Support](https://www.jaspersoft.com/support) (<https://www.jaspersoft.com/support>) can analyze this file.

To use the `--debugtrace` option, run the installer from the command line and specify an output filename. The precise command depends on your platform (Linux, Windows, or Mac OSX). For example, you can execute the installer with a command similar to the following:

```
jasperreports-server-9.0.0_linux_x86_64.run --debugtrace install-trace-out.bin
```

When you run the installer in `--debugtrace` mode, the installer takes extra time to write the binary output file. The final size of the output file is approximately 10 mg. Contact [Jaspersoft Technical Support](https://www.jaspersoft.com/support) (<https://www.jaspersoft.com/support>) to hand off the binary file for analysis.

Error Running Buildomatic Scripts

The buildomatic scripts depend on both Java and Apache Ant. Two common configuration errors are possible when attempting an installation using these scripts (if you're not using

the included, bundled Apache Ant).

Missing Java JDK

If you have the Java JRE (Java Runtime Environment) instead of the JDK, you won't have all the required utilities. In particular, you may see an error referring to the tools.jar, as in the following message:

```
[exec] [ERROR] BUILD FAILURE
[exec] [INFO] -----
[exec] [INFO] Compilation failure
[exec] Unable to locate the Javac Compiler in:
[exec]   c:\Program Files\Java\jdkx.x.x_xx\jre\..\lib\tools.jar
[exec] Please ensure you are using JDK x.x or above and
[exec] not a JRE (the com.sun.tools.javac.Main class is required).
[exec] In most cases you can change the location of your Java
[exec] installation by setting the JAVA_HOME environment variable.
```

The solution is to download and install the Sun Java JDK, labeled as the Java SE Development Kit on the Oracle web site.

Forgot to Copy the File ant-contrib.jar

If you're using your own version of Ant and your Ant instance doesn't have the ant-contrib.jar in the lib directory, you'll get an error similar to the following:

```
BUILD FAILED
c:\js-builds\jasperserver\buildomatic\install.xml:6:
```

Ant failed to create a task or type. To correct the error, copy <js-install>/buildomatic/extra-jars/ant-contrib.jar to your <apache-ant>/lib directory.

Failure with '\$' Character in Passwords in Buildomatic Scripts

If your password in buildomatic scripts includes two or more '\$' characters in a row, Ant will not accept it. This issue does not occur when dollar signs are separated by other characters. For example, \$pa\$password\$ or pa\$password\$ will not fail.

If you have two consecutive dollar signs, you'll need to escape each with three more dollar signs. For example, if your password is pa\$password, enter it as pa\$\$\$\$\$\$\$\$word in the configuration file. Once you do this, JasperReports Server will set all data connections to pa\$password.

Older Apache Ant Version

We recommend Apache Ant version 1.10. The earliest compatible version is Ant 1.9.

Older versions of Ant will cause an error similar to the following:

```
BUILD FAILED
c:\js-builds\jasperserver\buildomatic\install.xml:37:
Problem: failed to create task or type componentdef
```

To check your version of Ant and verify that it's at a high enough level, enter:

```
ant -version
```

If you have an earlier version of Ant, check to see if it's set in your class path by entering:

```
echo %CLASSPATH%
```

To use the JasperReports Server version of Ant, update your CLASSPATH variable to point to the <js-install>/apache-ant/bin directory.

Unable to Edit Files on Windows 10

In some cases, you may want to edit files manually in your C:/Jaspersoft directory during or after installation. For security reasons, Windows 10 does not allow normal processes to

change files in many folders, including the Program Files folder, for instance. When you attempt to edit these files, you may see an error like this:

```
You don't have permission to save in this location. Contact the
administrator to obtain permission.
```

You can edit these files by running as administrator. For example, to edit these files with Notepad on Windows 10:

Click Start and find the Notepad application. Right-click Notepad and click Run as administrator.

Bash Shell for Solaris, IBM AIX, HP UX and FreeBSD

The bash shell is required to execute the `js-install` shell scripts described in [Installing the WAR File for Production](#). The following `js-install` and `js-upgrade` scripts are in the `buildomatic` folder:

```
js-install.sh
js-upgrade-newdb.sh
js-upgrade-samedb.sh
```

The bash shell is not included by default in all Unix platforms. When the bash shell is not available, you'll need to download and install the bash shell specific to your platform.

Alternatively, you can manually run the same “buildomatic” Ant targets that are run by the `js-install` script. These Ant targets are listed in [Troubleshooting Your Server Configuration](#).

Also, make sure you've updated your local Ant to include `ant-contrib.jar`, which supports conditional logic in Ant. Copy the `ant-contrib.jar` to your `<ant_home>/lib` folder from:

```
buildomatic/extra-jars/ant-contrib.jar.
```

For more information see [Forgot to Copy the File ant-contrib.jar](#).

If you try using the Ant that's included with the JasperReports Server WAR file Distribution ZIP package, you may get the same non-bash syntax error. You may get the error below, for example:

```
js-ant help-install
ANT_HOME=../apache-ant: is not an identifier
```

If you have the bash shell installed, you can try executing the js-ant command by calling bash explicitly, for example:

```
bash js-ant help-install
```

Linux Issues

File Issues with Extended Character Sets on Linux

Your operating system configuration can influence the behavior of characters supported by JasperReports Server. On some Linux systems, Oracle JDK 8 sets the 'sun.jnu.encoding' system property to define the character set for encoding file names for I/O operations. This system property is set up on Java application startup and takes its value from the Linux system locale.

However, if the operating system is configured to use a non-UTF-8 encoding, JasperReports Server components may function in an unexpected way. For example, log collectors with non-UTF-8 names may be configured incorrectly. To fix these problems on a Linux operating system, enable Unicode support by setting the LANG and LC_ALL environment variables to a locale with the UTF-8 character set. This allows the operating system to process any character in Unicode. For example, LANG=en_US.UTF-8 and LC_ALL=en_US.UTF-8.

Linux Installer Issue with Unknown Host Error

If your Linux server doesn't have proper hostname entries in the /etc/hosts file, you may get installer errors.

The installer carries out an import operation to load the core minimal data into the repository database. This import operation can fail if the host is not configured.

If the import operation fails during installation, the installation will also fail. However, there should be an `installation.log` in the root of the installation folder to help debug the problem. The `installation.log` is located here:

```
<js-install>/installation.log
```

An improperly configured hosts file typically causes error messages like these:

```
Caused by: java.net.NoRouteToHostException: No route to host
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications
link failure
ERROR Cache:145 - Unable to set localhost. This prevents creation of a GUID
java.net.UnknownHostException
org.quartz.SchedulerException: Couldn't get host name!
```

To fix the `/etc/hosts` file

1. Include entries that look like these:

```
127.0.0.1      localhost.localdomain
172.17.5.0    myhost.mydomain.com      myhost
```

For instance:

```
127.0.0.1      localhost.localdomain      localhost
172.17.5.0    myhost.jaspersoft.com      myhost
```

2. You can also double check the file `/etc/sysconfig/network` (if it exists). In this file it would be similar to the following:

```
HOSTNAME=myhost
```

3. After fixing the `/etc/hosts` file, reinstall JasperReports Server.

Installation Error with Windows Path

If the path of the war archive exceeds the maximum length allowed by Windows, you'll get an error message like the one shown below.

```
java.io.IOException: Cannot run program "C:\Program
Files\Java\jdkx.x.x_xx\jre\bin\java.exe": CreateProcess error=206, The
filename
or extension is too long
    at java.lang.ProcessBuilder.start(ProcessBuilder.java:460)
    at java.lang.Runtime.exec(Runtime.java:593)
```

You'll need to move the war archive to reduce the path length. More information is available from Microsoft at:

[http://msdn.microsoft.com/en-us/library/windows/desktop/aa365247\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx).

Mac OSX Issues

Problem Starting JasperReports Server on Mac

We have seen some issues caused by the improper shutdown of the Tomcat included with JasperReports Server. This may be caused by shutting the machine down while Tomcat is running.

When the Tomcat scripts start Tomcat, they write a .pid (Process ID) file to the Tomcat folder. Tomcat uses this to determine whether the Tomcat instance is already running. When Tomcat is shutdown, this pid file is removed. However, if the pid file is not removed on shutdown, Tomcat will fail to start up.

You may see this when you double-click the jasperServerStart.app startup. JasperReports Server seems to be starting up, but it never actually does.

To recover from this issue, manually delete the pid file.

Delete catalina.pid using Finder

1. Navigate to the <js-install>/tomcat/temp folder
For instance: /Applications/jasperreports-server-9.0.0/tomcat/temp
2. Delete catalina.pid

Delete the catalina.pid file using Terminal shell

1. Open a Terminal shell (Finder > Go > Utilities > Terminal Icon)
2. Navigate to the <js-install>/tomcat/temp folder
For instance: /Applications/jasperreports-server-9.0.0/tomcat/temp
3. Enter the following command:
`rm catalina.pid`

To start and stop the PostgreSQL and Tomcat components separately from the command line shell

1. Open a Terminal shell (Finder > Go > Utilities > Terminal Icon).
2. Navigate to the <js-install> folder.
For instance: /Applications/jasperreports-server-9.0.0
3. To Start:
`./ctlscript start postgresql`
`./ctlscript start tomcat`
4. To shutdown:
`./ctlscript stop`
or
`./ctlscript stop tomcat`
`./ctlscript stop postgresql`

Database-related Problems

Database Privileges Required By JasperReports Server

Install/upgrade process permissions

The JasperReports Server installation/upgrade processes and the repository database user need the following privileges to install and initialize the jasperserver repository database.

Database Component	Permissions Required
databases	CREATE DROP
tables	CREATE
indexes	ALTER
constraints	DROP
data records	INSERT UPDATE DELETE

If you are upgrading in a restricted environment, your database administrator may need to give you temporary admin permissions for the upgrade. For example, if you are using PostgreSQL for your database, the database admin may use one of the following workarounds:

- Add administrator credentials in the `default_master.properties` file prior to upgrade and then replace them with `jasperadmin` credentials after upgrade.
- Prior to upgrade, grant `CREATE` and `DROP` permissions at the database server level for the `jasperadmin` user, then revoke those permissions after successful upgrade.

Database Connectivity Errors

The most common problems encountered with a new JasperReports Server instance are database configuration problems. The connection may fail because the application server cannot find the driver for the data source. For example, in a default installation of JasperReports Server, Tomcat looks for data source drivers in `<js-install>/apache-tomcat/lib`. If the driver's in a different location, put a copy of the driver in this directory and restart Tomcat.

Testing the Database Connection

The simplest database configuration problem is an incorrect username or password. If you encounter database problems on startup or login, check the username and password by

logging directly into your RDBMS as described below.

You can connect to your database using the database configuration settings in JasperReports Server. This validates the database hostname, port, username, and password.

If you are having trouble logging into JasperReports Server on the login page, check the existing users and passwords in the `jasperserver.JIUser` table.

Logging into PostgreSQL

Run the PostgreSQL client from the command line and try to connect to the database. For example:

```
psql -U postgres jasperserver
```

Logging into MySQL

Run the MySQL client from the command line and try to log in directly as the `root` user, for example:

```
<mysql>/bin/mysql -u root -p
```

You are prompted for the password of the user that you specified on the command line.

Logging into Oracle

Start SQL*Plus and try logging into Oracle directly. Use the password specified during installation to log in as each of these users:

- `jasperserver` — schema user for the JasperReports Server metadata.
- `sugarcrm` — schema user for the SugarCRM sample data.
- `foodmart` — schema user for the foodmart sample data.

Logging into Microsoft SQL Server

Run the `sqlcmd` and try logging into MSSQL Server directly. For example:

```
sqlcmd -S localhost\jasperserver -d jasperserver -U jasperadmin -P  
password
```


Connectivity Errors with SQL Server Driver

If you are using the SQL Server driver and have configured `default_master.properties` as described in [SQL Server Example](#), you'll see connection errors if you uncommented the following line:

```
# admin.jdbcUrl=jdbc:sqlserver://${dbHostOrInstance};SelectMethod=cursor
```

Make sure that this line is commented.

Case-sensitive Collation in SQL Server

Microsoft SQL Server does not support standalone case-sensitive collation. When collation is case-sensitive SQL Server also treats column and table names as case-sensitive. This can happen when setting a locale that includes case-sensitive collation. In this case you may see an error such as the following.

```
[sql] Failed to execute:
INSERT INTO JIUserRole (userId,roleId) select u.id, r.id
from JIUser u, JIRole r
where u.username = \'anonymousUser\' and r.roleName = \'ROLE_ANONYMOUS\'
[sql] com.microsoft.sqlserver.jdbc.SQLServerException: Invalid column
name \'roleName\'
```

Use a different locale or remove the case-sensitivity setting.

Configuring the Oracle or SQL Server Driver for NTLM Authentication

To avoid storage of the user and password values for the database, you can configure the Oracle or SQL Server driver to use Windows authentication. To do this, copy the NTLM authentication DLLs for your database from the `<js-install>\jasperserverwar\tools` directory to a location in your Windows system path (defined by the `PATH`

environment variable) and onfigure them as described in the instructions for your database:

- <https://blogs.oracle.com/blogbypuneeth/post/steps-to-configure-kerberos-spnego-ntlm-authentication-with-weblogic-server-running-on-oracle-jdk->
- <https://learn.microsoft.com/en-us/sql/connect/jdbc/using-ntlm-authentication-to-connect-to-sql-server?view=sql-server-ver16>

Maximum Packet Size in MySQL

If you are upgrading or importing into a MySQL database and your repository contains large objects like images, you may see an error like this:

```
ERROR 1153 (08S01): Got a packet bigger than 'max_allowed_packet' bytes
```

The default `max_allowed_packet` on the MySQL server is 1M (one Megabyte = 1,048,576 bytes). The most effective fix is to change this value in the server configuration to accommodate the largest resource stored in your repository. The server configuration file is typically named `my.cnf` (or `my.ini`) and located in the MySQL root directory, but this may vary. Change the configuration setting to a larger value, for example:

```
max_allowed_packet = 16M
```

For more information, see <http://dev.mysql.com/doc/refman/5.0/en/packet-too-large.html>.

After changing this value, restart the MySQL server. Then perform the upgrade or import step again.

Connection reset by peer MySQL Error

If you are using the MariaDB JDBC driver to connect to the MySQL database and get an error such as the following:

```
Could not send query:  
Connection reset by peer: socket write error
```

This message refers to the maximum packet size error described above. Follow those instructions.

Case Sensitivity for Table and Column Names

In some databases, table names are case-sensitive and “customer” and “Customer” are two different tables.

If you are using a case-sensitive database for JasperReports Server, the table names specified in query strings in the JRXML file of a saved report must match the table names used in the database. A mismatch when transferring data from one database to another may cause the capitalization of table names to change.

In Windows MySQL, table and column names are not case-sensitive.

In Linux MySQL, table and column names are case-sensitive. You can configure Linux MySQL to be non-case-sensitive by setting the configuration parameter `lower_case_table_names` to 1 in the `my.ini` or `my.cnf` file. For more information, search the MySQL documentation for a section about identifier case sensitivity.

Table and column names in Oracle and PostgreSQL are case-sensitive.

PostgreSQL: Job Scheduling Error

If the Quartz settings in the PostgreSQL database are not updated to specify the driver delegate class specific to PostgreSQL, then you get errors when you try to run a scheduled report.

The errors look like this:

```
Error while fetching Quartz runtime information
org.quartz.JobPersistenceException: Couldn't obtain triggers: Bad value for
type int
org.postgresql.util.PSQLException: Bad value for type int
```

If you see this error, check your Quartz properties file in the following location:

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

Make sure that the following property does not have the standard driver delegate, but instead has the PostgreSQL-specific driver delegate. It should look like the following for PostgreSQL:

```
quartz.delegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
```

Performance Issues with Oracle JDBC Queries

Setting the Oracle database localization option `defaultNChar` to `true` can substantially impact the performance of JDBC queries. When `defaultNChar` is set to `true`, the database implicitly converts all `CHAR` data to `NCHAR` when you access `CHAR` columns. If you do not need to support UTF-8 for your Oracle database, you can omit this setting.

The option you need and how to set it depends on your version of Java, your application server, and how it is deployed. For information about changing a JVM option setting for your particular environment, see your application server documentation.

To change this setting on Windows, enter a command like this at the command line:

```
set JAVA_OPTS=%JAVA_OPTS% -Doracle.jdbc.defaultNChar=false
```

To change this setting on Linux, enter a command like this at the command line:

```
export JAVA_OPTS="$JAVA_OPTS -Doracle.jdbc.defaultNChar=false"
```

Using an Oracle Service Name

If your Oracle database is configured to use a service name instead of an Oracle system identifier (SID), set up the service name by updating your `default_master.properties` file before using `buildomatic`:

```
<js-install>/buildomatic/default_master.properties
```

In `default_master.properties`, uncomment the `serviceName` property and enter your Oracle service name, for example:

```
serviceName=ORCL
```

When you are using an Oracle service name, make sure that you do not set the `SID` or `dbPort` in the `default_master.properties` file.

Error Running a Scheduled Report

If you run a scheduled report and save it as HTML or RTF, the resulting report may be quite large. If you are running MySQL and get the error shown here, the problem may be the default size of the MySQL blob datatype.

```
JDBC exception on Hibernate data access  
org.hibernate.exception.GenericJDBCException: could not insert
```

You can increase the size of this datatype by updating your my.ini or my.cnf MySQL configuration file with the following setting:

```
max_allowed_packet=32M
```

Error Running a Report

If you can log into JasperReports Server but encounter an error when running a report, browse the repository to identify and resolve the problem.

One common problem with an individual report is the data source. To validate a data source connection:

1. Log into JasperReports Server as a user with administrative permissions and locate the report unit that returns errors.
2. Select the report and click the Edit button in the toolbar and identify the data source on the fourth edit page.
3. Edit the data source in the repository and check its settings.
4. Click the Test Connection button.

If the connection fails, perhaps the application server can't find the driver for the data source. For example, in a default installation of JasperReports Server, Tomcat looks for data source drivers in <js-install>/apache-tomcat/lib.

5. Test your report. If it still returns errors, edit the data source again and try checking other values, like the port used by the database.

Save Error with DB2 Database

When the DB2 database is your repository database, you may get errors when saving longer strings (over 50 characters) to data entry fields in the UI. For example, saving a resource with a name over 50 characters may cause an error like this:

```
Expected status code is 200, but was 400. Response body contained:  
An unexpected exception has occurred
```

The problem here is that DB2 handles UTF-8 characters differently than other Jaspersoft certified databases. When DB2 is used as the repository database, it limits the number of characters that can be entered in UI fields. The database columns holding these strings need to be made larger.

BeanDefinitionStoreException with DB2 Driver

When using the DB2 driver, you need to add properties manually to default_master.properties, or you get an error like the following.

```
[java] Resource name: applicationContext-virtual-data-source.xml  
[java] org.springframework.beans.factory.BeanDefinitionStoreException:  
Invalid bean definition with name 'dataSource' defined in file  
[/opt/JasperReports-Server-9.0.0-src/jasperserver/buildomatic/conf_  
source/iePro/applicationContext-export-config.xml]:  
Could not resolve placeholder 'dbPort' in string value
```

Add the following properties to your default_master.properties, setting the correct values for your installation:

```
db2.driverType=4  
db2.fullyMaterializeLobData=true  
db2.fullyMaterializeInputStreams=true  
db2.progressiveStreaming=2  
db2.progressiveLocators=2  
dbPort=50000  
js.dbName=JSPRSRVR  
sugarcrm.dbName=SUGARCRM  
foodmart.dbName=FOODMART
```

JDBC Driver Loading Error on Import/Export from WebLogic or WebSphere

If you are using WebLogic or WebSphere and want to run import/export from the command line, you need to manually copy the JDBC driver to the same location as the import/export scripts. If you have not copied these files, you may encounter the following error:

```
Cannot load JDBC driver class <database class>
```

To fix this error, copy your database driver to the correct location:

```
from:      <js-install>\buildomatic\conf_source\db\<your_database>\jdbc\  
to:        <js-install>\buildomatic\conf_source\iePro\lib
```

Application Server-related Problems

Memory Issues Running Under Tomcat

These steps might solve problems related to the release of memory or to container tag pooling:

1. Set the following parameter in the global `$CATALINA_BASE/conf/web.xml`:
`enablepooling = false`
2. Restart Tomcat.

Java Out of Memory Error

If you encounter a Java out of memory error, try increasing your Java heap size setting. See [Setting JVM Options for Application Servers](#). As a minimum, add `-Xms2048m -Xmx4096m` to your `JAVA_OPTS` setting. You may need to increase this setting according to your usage.

This Java option is set within the application server, so you must restart your application server.

JVM Crash

Some combinations of application server and java builds under intense load can crash with an error like the following:

```
# A fatal error has been detected by the Java Runtime Environment:
#
# SIGSEGV (0xb) at pc=0x00007f408f0bf43b, pid=16819, tid=16848
#
# JRE version: OpenJDK Runtime Environment 18.9 (11.0.11+9) (build
11.0.11+9-LTS)
# Java VM: OpenJDK 64-Bit Server VM 18.9 (11.0.11+9-LTS, mixed mode,
sharing, tiered, compressed
oops, g1 gc, linux-amd64)
# Problematic frame:
# V [libjvm.so+0x7dd43b] G1ParCopyClosure<(G1Barrier)0, (G1Mark)0>::do_
oop(unsigned int*)+0x5b
#
# Core dump will be written. Default location: Core dumps may be
processed with
"/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h %e" (or dumping
to /opt/apache-tomcat9.0.37/core.16819)
#
# An error report file with more information is saved as:
# /tmp/hs_err_pid16819.log
#
# If you would like to submit a bug report, please visit:
# https://bugzilla.redhat.com/enter_
bug.cgi?product=Red%20Hat%20Enterprise%20Linux%208&component=java-11-
openjdk
```

To resolve this issue, you need to set the following additional Java properties to JAVA_OPTS setting:

```
set JAVA_OPTS=%JAVA_OPTS% -XX:+UseG1GC -XX:+ExplicitGCInvokesConcurrent -
XX:+ParallelRefProcEnabled -XX:+UseStringDeduplication -
XX:+UseCompressedClassPointers -XX:+UseCompressedOops
```

This Java option is applied to Java 8 and Java 11.

Configuration File Locations

You find JasperReports Server configuration properties specific to your application server in the following files.

Tomcat:	<tomcat>/webapps/jasperserver-pro/META-INF/context.xml
	<tomcat>/webapps/jasperserver-pro/WEB-INF/classes/hibernate.properties
	<tomcat>/apache-tomcat/webapps/jasperserver-pro/WEB-INF/web.xml (JNDI config)
	<tomcat>/apache-tomcat/config/Catalina/localhost/jasperserver-pro.xml (delete: see below)

JBoss:	<jboss>/standalone/deployments/jasperserver-pro.war/WEB-INF/js-jboss7-ds.xml
	<jboss>/standalone/deployments/jasperserver-pro.war/WEB-INF/classes/hibernate.properties
	<jboss>/standalone/deployments/jasperserver-pro.war/WEB-INF/web.xml
	<jboss>/standalone/deployments/jasperserver-pro.war/WEB-INF/jboss-web.xml

Tomcat Installed Using apt-get/yum

Setting CATALINA_HOME

If you are installing JasperReports Server to an instance of Tomcat that was installed using a package manager like apt-get, yum, or rpm, you can use the CATALINA_HOME and CATALINA_BASE properties found in your default_master.properties file.

Go to the section of the default_master.properties that looks like this:

```
# Tomcat app server root dir
appServerDir = C:\\Program Files\\Apache Software Foundation\\Tomcat 9.0
# appServerDir = /home/devuser/apache-tomcat-9.0
# if linux package managed tomcat instance, set two properties below
# CATALINA_HOME = /usr/share/tomcat9
# CATALINA_BASE = /var/lib/tomcat9
```

And change which lines are commented so it looks like this:

```
# Tomcat app server root dir
# appServerDir = C:\\Program Files\\Apache Software Foundation\\Tomcat
9.0
# appServerDir = /home/devuser/apache-tomcat-9.0
# if linux package managed tomcat instance, set two properties below
CATALINA_HOME = /usr/share/tomcat9
CATALINA_BASE = /var/lib/tomcat9
```

Note that you must set both CATALINA_HOME and CATALINA_BASE.

Database Driver Location

After installing JasperReports Server, make sure that there is a copy of the database driver file in the `/usr/share/tomcat9/webapps/jasperserver-pro/WEB-INF/lib` directory. If it is not there, copy the driver to this location. For example, for PostgreSQL, you can copy the driver from the `<js-install>/buildomatic/conf_source/db/postgresql/jdbc` directory.

JBoss Modifications

JBoss 7.2.0 Startup JDBC Version Error

The installation with JBoss EAP 7.2.0 may fail with the following error: "Detected Elasticsearch JDBC jar but cannot retrieve its version." In this case, you can remove the JAR file from the JasperReports Server WAR file so that the installation and startup can proceed.

1. Edit the WAR file with the following command on Linux:

```
zip -d jasperserver-pro.war WEB-INF/lib/x-pack-sql-jdbc-7.6.0.jar
```

You can verify that the JAR file has been removed with the following Linux command:

```
jar -tf jasperserver-pro.war | grep x-pack*
```

2. Stop the JBoss app server.

3. Delete the jasperserver-pro.war objects in the <jboss-eap-7.2>/standalone/deployments directory.
4. Switch to the JRS buildomatic directory and redeploy the JRS war file:


```
./js-ant deploy-webapp-pro
```
5. Restart the JBoss app server.

JBoss 7 Startup Timeout Error

JBoss 7 has a default startup time period. If your JBoss 7 takes longer than 60 seconds to start or deploy, you may receive the following error:

```
“(DeploymentScanner-threads - 1) Did not receive a response to the deployment operation within the allowed timeout period [60 seconds]. Check the server configuration file and the server logs to find more about the status of the deployment”.
```

To fix this, you need to increase your deployment-timeout setting as follows:

1. Change to the JBoss standalone configuration directory.


```
cd <jboss>/standalone/configuration
```
 2. Open the standalone.xml file.
 3. Look for the <subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1"> element, for example:


```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
  <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000"/>
</subsystem>
```
 4. Edit this to add or set the attribute deployment-timeout to the preferred time in seconds, for example:


```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
  <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000" deployment-timeout="600"/>
</subsystem>
```
 5. Save the file.
- On server restart, your system has the specified time to start up.

Using a Non-default JBoss Profile

When you set up the installation in the `JasperReports-server-pro-8.0.0-bin\buildomatic` folder, the `build_conf` folder will not be created. It is created when you run the `js-install` script. If JBoss is your application server and you are using a profile other than the default, then you need to set the `jboss7.profile` property before running the `js-install` script in [Installing the WAR File Using js-install Scripts](#):

1. Open the file `jasperreports-server-pro-8.0.0-bin/buildomatic/conf_source/templates/app.srv.properties`.
2. Change the profile name as follows:
`jboss 7 profile`
`jboss7.profile = <profile name>`

Using JBoss with Non-Latin Characters

If JBoss is your application server and your organization is created with non-Latin characters, then you need to edit the `standalone.xml` configuration file.

1. Edit `<jboss-home>/standalone/configuration/standalone.xml`
2. Add a new `<system-properties>` tag after the `<extensions>` tag, as shown in the following example.

```
<extensions>
.....
</extensions>
<system-properties>
<property name="org.apache.catalina.connector.URI_ENCODING" value="UTF-8"/>
<property name="org.apache.catalina.connector.USE_BODY_ENCODING_FOR_QUERY_STRING" value="true"/>
</system-properties>
```

Maximum Post Size in Wildfly

If you are upgrading or importing on some versions of Wildfly and your repository or other import file is large, the import may fail and the connection may be reset. In this case, you may need to set `max-post-size`. To do this, open the file `<wildfly-`

home>standalone/configuration/standalone.xml and add or change the `max-post-size` attribute of the `http-listener` property, for example:

```
<http-listener name="default" socket-binding="http" max-header-size="974247881"
    max-post-size="974247881"/>
```

WebSphere Modifications

Page Not Found Error on Login

You may see this error during a WebSphere installation when a user attempts to log into JasperReports Server. After typing in a correct user ID and password, the user sees an error page: Page cannot be found, HTTP 404

Some WebSphere versions or fix packs have modified code that processes web server filters incorrectly. Components with the `/*` URL pattern get affected by this. JasperReports Server uses the Spring framework for authentication and it is mapped using a filter chain with the `/*` URL pattern. You need to set a special property that WebSphere provides to solve this problem.

To solve the Page Not Found Error on Login

1. Login into the WebSphere Administrative Console.
2. Navigate to Application Servers > <server> > Web Container Settings > Web Container > Custom Properties.
3. Create a new property with the following attributes:
name: `com.ibm.ws.webcontainer.invokefilterscompatibility`
value: `true`
4. Save the master configuration.
5. Restart the WebSphere server.

WebLogic Modifications

Schema Validation Error

You may see an error like the following on some WebLogic installations:

```
<Critical> <WebLogicServer> <BEA-000362> <Server failed. Reason:
[Management:141245]Schema Validation Error in /u01/app/oracle/WLS/user_
projects/domains/x2o_uat_01/config/config.xml
```

This may be caused by the configuration of the `<stuck-thread-max-time>` element in the designated configuration file. In this case, removing `stuck-thread-max-time` may resolve the error.

Disabling User Session Persistence in Application Servers

JasperReports Server stores non-serializable data in its user sessions, which can cause errors after restarting your application server:

```
Exception loading sessions from persistent storage
Cause: java.io.NotSerializableException ...
```

The errors appear in the JasperReports Server log when users log in after the application server has been restarted. Users do not see the errors, and they have no impact on JasperReports Server operations.

Because JasperReports Server user sessions are not persistent, you can configure your application server to disable persistence and avoid the error. For example, in Apache Tomcat, edit the file `<tomcat>/conf/context.xml` and locate the following lines.

```
<!-- Uncomment this to disable session persistence across Tomcat
restarts -->
<!--
<Manager pathname="" />
-->
```

Remove the comment markers from lines 2 and 4 above, then restart Apache-Tomcat activate the change. For other application servers, refer to the product documentation.

License-related Errors

License Not Found Errors

Normally, the JasperReports Server installer includes an evaluation license file that you replace with a commercial license file, as described in [Installing a New License File](#). If JasperReports Server returns an error after you replace the license file, the most likely causes are:

- You didn't clear your application server's work directory, as explained in [Installing a New License File](#). Delete the work directory, restart the application server, and try logging into JasperReports Server again.
- The `Djs.license.directory` property in your application server startup environment is incorrectly set:

For example, in Windows the correct setting looks like this:

```
-Djs.license.directory=<js-install>
```

In Linux, the correct setting looks like this:

```
-Djs.license.directory=/opt/jasperreports-server-9.0.0
```

The specified directory must contain the license file, named `jasperserver.license`. The property is set in `JAVA_OPTS` in the `standalone.conf` file. It must contain the location of your license file, which is usually the `<js-install>` directory:

Tomcat: `<tomcat>/bin/setclasspath.bat/.sh` or `bin/setenv.bat/.sh`

JBoss: `<jboss>\bin\standalone.conf.bat` or `<jboss>/bin/standalone.conf`

License Not Found or License Corrupt Error with Tomcat as a Service

If you have Tomcat running as a service in Windows, the installer attempts to make the proper updates so the server license file is found at when you start the application server. If the installer is unsuccessful, make sure you took the steps described in [License File for Existing Tomcat as Windows Service](#).

Problems Importing and Exporting Data from the Repository

Exporting a Repository That Contains UTF-8

You may see the following errors when you have international characters in repository objects, for example, in user IDs.

Error During Export

An Upgrade usually requires exporting your database. If you're using MySQL and getting this null pointer exception, it may be caused by an incorrect character in the `js.jdbc.properties` file:

```
java.lang.NullPointerException
ResourceExporter.exportResource(ResourceExporter.java:258)
```

Check the URL in this file in `<js-install>buildomatic/build_conf/default/`; it should look like this:

```
jdbc:mysql://localhost:3306/jasperserver?useUnicode=true&characterEncoding=UTF-8
```

Note the ampersand `&`. It's incorrect if it appears as `&`. The `&` is correct only in an HTML or XML context. It's incorrect in a properties file.

Error During Export from Repository on Oracle

Oracle requires a specific JVM property to handle UTF-8 characters properly. If the export is empty and this error occurs when attempting to compress the result:

```
ERROR ExporterImpl:129 - java.util.zip.ZipException: ZIP file must have at least one entry
```

If you have stored your repository database on an Oracle RDBMS, modify the last line of both `<js-install>/buildomatic/js-export.*` files as follows:

```
From: java -classpath ...
To: java -Doracle.jdbc.defaultNChar=true -classpath ...
```


Problems with Upgrade

Include Audit Events on Upgrade

If you have auditing enabled and you run upgrade using `js-upgrade-newdb.bat/sh`, audit events are not imported by default. To import audit events, you need to run an additional command after the `js-upgrade-newdb` script completes. To do this, change to the `buildomatic` directory:

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

Then run one of the following commands:

```
js-import.bat --input-zip=js-my-export-all.zip --include-audit-events  
(Windows)
```

```
js-import.sh --input-zip=js-my-export-all.zip --include-audit-events  
(Linux)
```

or

```
ant import -DimportFile=js-my-export-all.zip -DimportArgs="--include-audit-events"
```

These commands re-import all resources from the specified export file, add the audit event, and do not overwrite existing resources.



When using either import utility, the server must be stopped to avoid issues with caches, configuration, and security.

Overlay Upgrade Permissions Error with Bundled Installation

If you are using the overlay upgrade procedure with a PostgreSQL database, and you installed an earlier version of JasperReports Server, you might see an HTTP Status 404 error.

The overlay installer is not supported with the bundled installation and may only be used with a war file installation of JRS.

Overlay Upgrade Domain Issue with MySQL and MariaDB JDBC Driver

When working with Domains using the MySQL database and using the 1.1.2 version of the MariaDB JDBC driver (mariadb-java-client-1.1.2.jar) there can be an issue handling Boolean values correctly. The fix is to upgrade to a higher version of the MariaDB JDBC driver. For supported versions of the driver, see the Jaspersoft Platform Support Guide.

Because the Overlay packages for some previous JasperReports Server versions were distributed with the MariaDB 1.1.2 JDBC driver, this error can also occur even without an upgrade operation.

The issue may occur when a Boolean filter is created on a domain. Because of a bug in older versions of the MariaDB driver, the Boolean filter is evaluated as a numeric filter.

<https://mariadb.atlassian.net/browse/CONJ-72>

Manually Creating the JasperReports Server Database

If you can't use the `js-install` scripts to create the JasperReports Server database and the sample databases, you can create them manually. Follow the instructions for your database to create the repository database and optional sample databases:

- [PostgreSQL](#)
- [MySQL](#)
- [Oracle](#)
- [DB2](#)
- [SQL Server](#)

The commands in these sections have been tested at Jaspersoft, but the commands you need to use on your database instance may be different.



For running the Ant commands, you need to edit the `default_master.properties` file to add the settings for your database and application server as described in [Installing the WAR File Using js-install Scripts](#).

PostgreSQL

To manually create the JasperReports Server database in PostgreSQL

1. On the Windows, Linux, or Mac command line, enter these commands:

```
cd <js-install>/buildomatic/install_resources/sql/postgresql
psql -U postgres -W
postgres=#create database jasperserver encoding='utf8';
postgres=#\c jasperserver;
postgres=#\i js-pro-create.ddl
postgres=#\i quartz.ddl
postgres=#\q
```

2. Run the following commands to install the JSAudit database:

```
cd <js-install>/buildomatic/install_resources/sql/postgresql
psql -U postgres -W
postgres=#create database jsaudit;
postgres=#\c jsaudit;
postgres=#\i js-pro-create-audit.ddl
postgres=#\q
```

3. (Optional) Run the following commands if you want to install sample databases:

```
cd <js-install>/buildomatic/install_resources/sql/postgresql
psql -U postgres -W
postgres=#create database sugarcrm encoding='utf8';
postgres=#create database foodmart encoding='utf8';
postgres=#\c sugarcrm;
postgres=#\i sugarcrm.sql; (first make sure the file is unzipped)
postgres=#\c foodmart;
postgres=#\i foodmart-postgresql.sql; (first make sure the file is
unzipped)
postgres=#\i supermart-update.sql;
postgres=#\q
```

4. If you didn't install the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic  
js-ant import-minimal-pro  
js-ant deploy-webapp-pro
```

If you installed the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic  
js-ant import-sample-data-pro  
js-ant deploy-webapp-pro
```

For more information about executing the Ant scripts, see [Installing the WAR File Manually](#).

5. Set Java JVM Options (required), as described in [Setting JVM Options for Application Servers](#).
6. Set up the JasperReports Server License (required) as described in [Setting Up the JasperReports Server License](#).

MySQL

To manually create the JasperReports Server database in MySQL

You can use the MySQL client software, `mysql.exe` or `mysql`, to interact with the MySQL database.



For specific details on connecting to the MySQL database and setting privileges for databases and db users, please refer to the documentation provided with your database.

-
1. On the Windows, Linux, or Mac command line, enter the following commands to create and initialize the JasperReports Server database.

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

```
mysql -u root -p
mysql>create database jasperserver character set utf8;
mysql>use jasperserver;
mysql>source js-pro-create.ddl
mysql>source quartz.ddl
mysql>exit
```

2. Run these commands to create and initialize the JSAudit database.

```
mysql -u root -p
mysql>create database jsaudit;
mysql>use jsaudit;
mysql>source js-pro-create-audit.ddl
mysql>exit
```

3. (Optional) Run these commands to install sample databases:

```
cd <js-install>/buildomatic/install_resources/sql/mysql
mysql -u root -p
mysql>create database sugarcrm;
mysql>create database foodmart;
mysql>use sugarcrm;
mysql>source sugarcrm.sql;(first make sure the file is unzipped)
mysql>use foodmart;
mysql>source foodmart-mysql.sql; (first make sure the file is unzipped)
mysql>source supermart-update.sql;
mysql>exit
```

4. If you didn't install the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic
js-ant import-minimal-pro
js-ant deploy-webapp-pro
```

If you installed the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic
js-ant import-sample-data-pro
js-ant deploy-webapp-pro
```

For more information about executing the Ant scripts, see [Installing the WAR File Manually](#).

5. Set Java JVM Options (required), as described in [Setting JVM Options for Application Servers](#).
6. Set up the JasperReports Server License (required) as described in [Setting Up the JasperReports Server License](#).

Oracle

To manually create the JasperReports Server database in Oracle

You can use the Oracle client software, `sqlplus.exe` or `sqlplus`, to interact with Oracle.



For specific details on connecting to the Oracle database and setting privileges for databases and db users, please refer to the documentation provided with your database.

1. On the Windows, Linux, or Mac command line, enter the following commands to create and initialize the JasperReports Server database.

```
cd <js-install>/buildomatic/install_resources/sql/oracle
sqlplus /nolog (start sqlplus client)
SQL> connect system/password (use your sysUsername and password)
(or SQL>connect sys/password as sysdba
SQL> create user jasperserver identified by password; (as sys user)
SQL> grant connect, resource to jasperserver; (as sys user)
SQL> grant unlimited tablespace to jasperserver; (as sys user)
SQL> connect jasperserver/password@ORCL (use your password, your SID)
```

```
SQL> @/opt/jasperreports-server-pro-8.0.0-bin/buildomatic/install_
resources/sql/oracle/js-pro-create.ddl
SQL> @/opt/jasperreports-server-pro-8.0.0-bin/buildomatic/install_
resources/sql/oracle/js-pro-create-audit.ddl
SQL> @/opt/jasperreports-server-pro-8.0.0-bin/buildomatic/install_
resources/sql/oracle/quartz.ddl
SQL> exit
```

2. To create and initialize the JSAudit database, enter the following commands.

```
SQL> create user jsaudit identified by password; (as sys user)
SQL> grant connect, resource to jsaudit; (as sys user)
SQL> grant unlimited tablespace to jsaudit; (as sys user)
SQL> connect jsaudit/password@ORCL
SQL> @/opt/jasperreports-server-pro-8.0.0-bin/buildomatic/install_
resources/sql/oracle/js-sequence-create.ddl
SQL> @/opt/jasperreports-server-pro-8.0.0-bin/buildomatic/install_
resources/sql/oracle/js-pro-create-audit.ddl
SQL> exit
```

3. Go to the <js-install>/buildomatic path and configure the default_ master.properties file with the required values. For example:

```
cd <js-install>/buildomatic
dbUsername=jasperserver
dbPassword=password
sysUsername=jasperserver
sysPassword=password
dbHost=localhost
dbPort=1521 sid=ORCL
```

```
#audit props
installType=split
audit.dbHost=localhost
audit.dbPort=1521
audit.sid=ORCL
audit.dbUsername=jsaudit
audit.dbPassword=password
audit.dbName=jsaudit
audit.sysUsername=system
audit.sysPassword=password
```


You can set `sysUsername` and `sysPassword` the same as `dbUsername` and `dbPassword`.

4. Create server setting with audit db schema name (`auditDB=JSAUDIT`), to do so:
 - a. Go to `<js-install>/buildomatic/bin` path and edit the `db-common.xml` file.
 - b. Add the following target at the end of file and before `</project>`:


```
<target name="import-profile-attributes">
  <import-profile-attribute key="auditDB"
    attrValue="\${audit.dbName}"/>
</target>
```
 - c. Save the file.
 - d. Run the following command:


```
./js-ant import-profile-attributes
```



Server setting `auditDB=JSAUDIT` is needed for audit reports working properly on oracle in case of split installation.

5. (Optional) Special edit to the `sugarcrm.sql` script that creates the `sugarcrm` sample database. The `sqlplus` command line tool interprets SQL statements differently than a JDBC call (that is, the way `buildomatic` executes SQL scripts). Because of this, the `sugarcrm.sql` file must be edited in order to execute using `sqlplus`. To make these edits do the following:
 - Unzip the `sugarcrm.zip` file to get the `sugarcrm.sql` file. Open `sugarcrm.sql` for editing:
 - Uncomment the `-- set define off` line to look like this `set define off` (Line 7)
 - Uncomment the `--/"` line that follows the `CREATE TRIGGER` statements (there are 12 of these toward the very end of the file on line 71,282. Just before the `CREATE INDEX` statements). Change to be just `/"`. (This terminates the trigger procedure definition in `sqlplus`.)
 - Save the file.



If you build and load the sample databases using `buildomatic`, the `NLS_LANG` setting is automatically handled via a JDBC driver setting.

If you load the sample databases using `buildomatic`, you won't need to set any variables or make any script edits.

6. (Optional) Set the NLS_LANG variable. The sugarcrm database has test data that requires a specific NLS_LANG setting in order to load into Oracle correctly. You will need to set this in your shell environment if you're manually loading the sugarcrm database.

Windows:

```
set NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1
```

Linux:

```
export NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1
```

7. (Optional) Run the following commands if you want to install sample databases:

```
cd <js-install>/buildomatic/install_resources/sql/oracle
sqlplus /nolog (start sqlplus client)
SQL> connect system/password (use your sysUsername and password)
(or SQL>connect sys/password as sysdba
SQL> create user sugarcrm identified by password;
SQL> create user foodmart identified by password;
SQL> grant connect, resource to sugarcrm;
SQL> grant connect, resource to foodmart;
SQL> connect sugarcrm/password@ORCL
SQL> @sugarcrm.sql (First, make sure file is unzipped)
SQL> connect foodmart/password@ORCL
SQL> @foodmart-oracle.sql (First, make sure file is unzipped)
SQL> @supermart-update.sql
SQL> exit
```

8. If you didn't install the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic
js-ant import-minimal-pro
js-ant deploy-webapp-pro
```

If you installed the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic
js-ant import-sample-data-pro
js-ant deploy-webapp-pro
```

For more information about executing the Ant scripts, see [Installing the WAR File Manually](#).

9. Set Java JVM Options (required), as described in [Setting JVM Options for Application Servers](#).
10. Set up the JasperReports Server License (required) as described in [Setting Up the JasperReports Server License](#).

DB2

To manually create the JasperReports Server database in DB2

Use the DB2 client software, `db2` or `db2cmd`, to interact with DB2.



For specific details on connecting to the DB2 database and setting privileges for databases and db users, please refer to the documentation provided with your database.

1. Change to the following directory:

```
cd <js-install>/buildomatic/install_resources/sql/db2
```

2. Enter these commands in the DB2 command window to create and initialize the repository database called `jsprsrvr` in DB2 to conform to the 8-character limitation:

```
db2 create database jsprsrvr using codeset utf-8 territory us pagesize
16384
db2 connect to jsprsrvr
db2 -tf js-pro-create.ddl
db2 -tf quartz.ddl
```

3. To create and initialize the JSAudit database, enter the following commands in the DB2 command window:

```
db2 create database jsaudit
db2 connect to jsaudit
db2 -tf js-pro-create-audit.ddl
db2 exit
```

4. (Optional) Run the following commands in the DB2 command window if you want to install sample databases:
-

```
db2 create database sugarcrm
db2 connect to sugarcrm
db2 -tf sugarcrm.sql (first make sure file is unzipped)
db2 create database foodmart
db2 connect to foodmart
db2 -tf foodmart-db2.sql (first make sure file is unzipped)
db2 -tf supermart-update.sql (if script is available)
```

5. If you didn't install the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic
js-ant import-minimal-pro
js-ant deploy-webapp-pro
```

If you installed the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic
js-ant import-sample-data-pro
js-ant deploy-webapp-pro
```

For more information about executing the Ant scripts, see [Installing the WAR File Manually](#).

6. Set Java JVM Options (required), as described in [Setting JVM Options for Application Servers](#).
 7. Set up the JasperReports Server License (required) as described in [Setting Up the JasperReports Server License](#).
-

Further considerations:

- If JasperReports Server is deployed on the same host as DB2, delete the following file to avoid conflicts:
<db2>/SQLLIB/java/db2jcc.jar

SQL Server

Use the sqlcmd utility to manually build the jasperserver database.



For specific details on connecting to the SQL Server database and setting privileges for databases and db users, please refer to the documentation provided with your database.

To manually create the JasperReports Server database in SQL Server

1. Open a Command Prompt and enter the following commands using the administrator (sa) user name and password.

```
cd <js-install>\buildomatic\install_resources\sql\sqlserver
sqlcmd -S ServerName -Usa -Psa
1> CREATE DATABASE [jasperserver]
2> GO
1> USE [jasperserver]
2> GO
1> :r js-pro-create.ddl
2> GO
1> :r quartz.ddl
2> GO
```

2. From the Windows Start menu, select Microsoft SQL Server > SQL Server Management Studio.
3. Connect to SQL Server as the administrative database user, and check that the jasperserver database appears in the Object Explorer.
4. Expand Tables in the jasperserver database, and check that the tables have been added.

To create and initialize the JSAudit database

5. Run the following commands:

```
cd <js-install>\buildomatic\install_resources\sql\sqlserver
sqlcmd -S ServerName -Usa -Psa
1> CREATE DATABASE [jsaudit]
2> GO
1> USE [jsaudit]
2> GO
1> :r js-pro-create-audit.ddl
2> GO
```

To manually create the optional sample databases in SQL Server

6. Extract the files in the sugarcrm.zip file to the level above your current directory, placing the sugarcrm.sql file in this directory:

```
<js-install>\jasperserver\buildomatic\install_resources\sql\sqlserver
```

7. Enter these commands to create and initialize the sugarcrm database:

```
1> CREATE DATABASE [sugarcrm]
2> GO
1> USE [sugarcrm]
2> GO
1> :r sugarcrm.sql
2> GO
```

8. You cannot initialize the foodmart database manually. Instead, change to the buildomatic directory and use the following buildomatic commands to create and initialize it from the command line:

```
js-ant create-foodmart-db  
js-ant load-foodmart-db
```

Alternatively, you can replace the first command and create the database manually using the following SQL Server commands, but you still have to use the buildomatic command `js-ant load-foodmart-db` to load the data:

```
1> CREATE DATABASE [foodmart]  
2> GO  
1> USE [foodmart]  
2> GO
```

To complete the manual installation of databases in SQL Server

9. If you didn't install the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic  
js-ant import-minimal-pro  
js-ant deploy-webapp-pro
```

If you installed the optional sample databases, complete the installation with these commands:

```
cd <js-install>/buildomatic  
js-ant import-sample-data-pro  
js-ant deploy-webapp-pro
```

For more information about executing the Ant scripts, see [Installing the WAR File Manually](#).

10. Set Java JVM Options (required), as described in [Setting JVM Options for Application Servers](#).
11. Set up the JasperReports Server License (required) as described in [Setting Up the JasperReports Server License](#).

Jaspersoft Documentation and Support Services

For information about this product, you can read the documentation, contact Support, and join Jaspersoft Community.

How to Access Jaspersoft Documentation

Documentation for Jaspersoft products is available on the [Product Documentation website](#), mainly in HTML and PDF formats.

The [Product Documentation website](#) is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the [JasperReports® Server Product Documentation](#) page.

How to Access Related Third-Party Documentation

When working with JasperReports® Server, you may find it useful to read the documentation of the following third-party products:

How to Contact Support for Jaspersoft Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our [product Support website](#).
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the [product Support website](#). If you do not have a username, you can request one by clicking **Register** on the website.

How to Join Jaspersoft Community

Jaspersoft Community is the official channel for Jaspersoft customers, partners, and employee subject matter experts to share and access their collective experience. Jaspersoft Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from Jaspersoft products. In addition, users can submit and vote on feature requests from within the [Jaspersoft Ideas Portal](#). For a free registration, go to [Jaspersoft Community](#).

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

Jaspersoft, JasperReports, Visualize.js, and TIBCO are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at <https://www.tibco.com/patents>.

Copyright © 2005-2024. Cloud Software Group, Inc. All Rights Reserved.