JasperServer War File Install Guide

Version 2.0.1

Table of Contents

1 Introduction

This document covers the installation of JasperServer if you do not use the installation binaries.

JasperServer is released as a "stand-alone War" file packaged as a zip file. A "War" file is a Web Application Archive. It is designed to be dropped into a web container application such as Apache Tomcat.

The WAR file - JasperServer-x.x.x-bin.zip, contains the Web UI interface and the Web Services (SOAP) interface.

2 Web UI interface

This section describe the steps for checking dependent components, installing JasperServer, setting up databases, configuring JasperServer for databases, and running JasperServer.

2.1 Component Dependencies

The JasperServer application was developed and tested using the following components. More configurations are being added and tested, but JasperServer is known to work with these configurations:

- Linux
  - Tomcat 5.5.12
  - MySQL 5.0.19
- Windows
  - Tomcat 5.5.12
  - MySQL 5.0.18

2.2 Getting the Installation Components

All JasperServer application components are currently available from the download area of the SourceForge site (http://jasperintel.sourceforge.net).

In the download area, you will find the JasperServer-<ver>.zip. This is the JasperServer stand-alone war file that you will need to download.

2.3 Pre-installing Dependent Components

Before you can run JasperServer, you will need to install or otherwise have the dependent components available. These components are Tomcat and MySQL.

To install Tomcat (http://tomcat.apache.org/) and/or MySQL (http://www.mysql.com/) please see the project web sites for these components and follow the instructions for installation.

2.4 Install JasperServer to Tomcat

Because the JasperServer stand-alone war package is a "war" file (Web Application Archive) the installation process involves dropping the war into the webapps directory of the Tomcat server.

You will need to first locate your Tomcat installation directory and locate the webapps directory:

- Linux:

  <tomcat-install-dir>/webapps

  Examples:

/opt/tomcat/webapps

- Windows:

  <tomcat-install-dir>/webapps

  Examples:

  C:\Tomcat\webapps, or

  C:\Program Files\Apache Tomcat\webapps

After unpacking the JasperServer package zip file, drop the jasperserver.war into the tomcat webapps directory:

- Copy:

  <jasperserver-unzipped-dir>/jasperserver.war

- To:

  <tomcat-install-dir>/webapps

2.5 Create Databases Using DB Scripts

The JasperServer war file download zip includes a "scripts" directory.

The scripts directory has the following scripts:

- <JasperServer-unzipped-dir>/scripts:
  - jasperserverCreate.ddl
  - jasperserverCreateDefaultSecurity.sql
  - upgrade scripts

2.5.1 Create the Database and DB User

Check your database user manual for how to set up a database and how to create a database user. You will need to create the following (suggested) databases and user names and passwords. With the MySQL database, enter the following commands:

```
mysql -u root -p        (login to mysql)

mysql> create database jasperserver; (for JasperServer metadata)

mysql> create database foodmart; (for JasperAnalysis sample data)

mysql> create database sugarcrm; (for JasperAnalysis sample data)
```

You can access the database from JasperServer using the mysql default user "root". Or you can create a database user specifically for JasperServer (as the JasperServer installers do). You can create this user using a mysql command similar to the following:

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

mysql> quit (exit MySQL)
```

2.5.2 Minimal Database Setup

If you want to run the JasperServer application without any test data, perform the following steps:

1. run the jasperserverCreate.ddl against your JasperServer database

2. run the jasperserverCreateDefaultSecurity against your JasperServer database

Examples for MySQL would be the following:

1. mysql -u root -p -h localhost –D jasperserver < jasperserverCreate.ddl
2. mysql -u root -p -h localhost –D jasperserver < jasperserverCreateDefaultSecurity.sql

2.5.3 Database Setup with Sample Data

JasperServer comes with an import/export utility for setting up sample data. Using this data you can run sample reports. To include this sample reporting functionality run the following scripts:

Under Windows:

```
ji-import.bat --input-dir ji-catalog
```

or under Linux:

```
ji-import.sh --input-dir ji-catalog
```

This command will create the sample reports and associated metadata for JasperReports. It will also create sample OLAP views for JasperAnalysis using the Foodmart and sugar CRM data created in 2.5.1.

2.5.4 Upgrading a JasperServer 1.1 metadata database

The scripts folder contains MySQL scripts that can be used to upgrade an older JasperServer metadata database to JasperServer 1.2.1.

Before running the scripts to upgrade the JasperServer database, it is recommended to take a backup of the database:

mysqldump --user=*user* --password=*password* jasperserver > jasperserver-dump.sql

There scripts/upgrade folder contains incremental upgrade scripts for each JasperServer release, starting with 1.1.0. One would hence need to run all the scripts (in order) corresponding to the version between the original JasperServer version and the current one.

For instance, to upgrade a 1.1.0 metadata database to 1.2.1, one would need to run

mysql --user=*user* --password=*password* -D jasperserver < jasperserverPatch-1.1.0-1.2.0.ddl

mysql --user=*user* --password=*password* -D jasperserver < jasperserverPatch-1.2.0-1.2.1.ddl

To upgrade a 1.2.0 metadata database to 1.2.1, one would only need to run

mysql --user=*user* --password=*password* -D jasperserver < jasperserverPatch-1.2.0-1.2.1.ddl

Afther the proper upgrade scripts are executed, the JasperServer 1.2.1 web application can be connected to the upgraded metadata database.

2.6 Connecting to the Databases (Unpacking War File)

The jasperserver.war file is "archived" so you cannot directly edit files inside of it. The JasperServer database configuration file is inside the war file, so we must "un-archive" the war file first.

Essentially, there are two ways to un-archive the jasperserver.war file:

1. Un-archive by hand using the java "jar" command

   (assumes you have the Java SDK).

2. Let Tomcat un-archive the war for you

2.6.1 Un-archive by hand using the java "jar" command

You can use the following steps to un-archive the war file:

1. cd <tomcat-install-dir>/webapps
2. mkdir jasperserver
3. cd jasperserver
4. jar xvf ../jasperserver.war

This will get all of the files and directories in the right ocations.

2.6.2 Let Tomcat un-archive the war for you

The following set of information is correct for Tomcat 5.0. If the jasperserver.war file is in the following directory:

```
cd <tomcat-install-dir>/webapps
```

Then, Tomcat will un-archive (as part of the war deployment) the war file as part of the Tomcat startup.

So, start Tomcat:

```
<tomcat-install-dir>/bin/startup.bat (or startup.sh)
```

In addition to un-archiving, Tomcat will copy the:

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

to:

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

Tomcat will use the above file for database configuration and not the META-INF/context.xml file. So, if you are doing database configuration, you will need to edit the file in the conf directory.

2.7 Connecting to the Databases (Edit Configuration File)

In order to connect to the databases you will need to modify the following configuration file:

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

or

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

You will need to set the username and password to your settings. Below is an example for the Sugar CRM example database:

```
<Context path="/jasperserver" docBase="jasperserver"
  <Resource name="jdbc/sugarcrm" auth="Container" type="javax.sql.DataSource"
          maxActive="100" maxIdle="30" maxWait="10000"
  username="<username>" password="<password>" driverClassName="com.mysql.jdbc.Driver"
  url="jdbc:mysql://localhost:3306/sugarcrm?autoReconnect=true"/>
  <Resource name="jdbc/jasperserver" auth="Container" type="javax.sql.DataSource"
          maxActive="100" maxIdle="30" maxWait="10000"
```

```
    username="<username>" password="<password>" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/jasperserver?autoReconnect=true"/>
</Context>
```

Below is an example for the Foodmart example database:

```
<Context path="/jasperserver" docBase="jasperserver"
    <Resource name="jdbc/MondrianFoodMart " auth="Container" type="javax.sql.DataSource"
            maxActive="100" maxIdle="30" maxWait="10000"
    username="<username>" password="<password>" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/foodmart?autoReconnect=true"/>
    <Resource name="jdbc/jasperserver" auth="Container" type="javax.sql.DataSource"
            maxActive="100" maxIdle="30" maxWait="10000"
    username="<username>" password="<password>" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/jasperserver?autoReconnect=true"/>
</Context>
```

Configuration files that reference the above jdbc settings are the following:

```
jasperserver/WEB-INF/web.xml
jasperserver/WEB-INF/hibernate.properties
```

If you did not install the sample database (Sugar CRM and Foodmart) then you can ignore the sugarcrm settings.

2.8 Starting the JasperServer Application

All databases should now have been created and JasperServer should be configured to access these databases.

JasperServer runs within a web container application such as Tomcat. In order to start the JasperServer application, you will start Tomcat.

To start the JasperServer Web Services application:

- Linux:

```
<tomcat-install-dir>/bin/startup.sh
```

- Windows:

```
<tomcat-install-dir>/bin/startup.bat (Tomcat 5.0), or
<tomcat-install-dir>/bin/tomcat5.exe (Tomcat 5.5)
```

You may check the tomcat runtime logs to see that there are not serious errors on the startup (note it is common to find "warnings" in the tomcat runtime logs):

```
<tomcat-install-dir>/logs/catalina-<date>.log
```

2.9 Login to JasperServer

If JasperServer started up properly, you should be able to login.

Login by going to the following URL:

http://<hostname>:8080/jasperserver

Example:

http://localhost:8080/jasperserver

The login page should appear (after some time to compile the necessary jsp files). Use the following ids to login to the system:

jasperadmin/newPassword	(Admin user - from minimal db setup)
tomcat/tomcat	(Admin user - from sample data)
joeuser/joe	(Normal user - from samle data)

If you have logged in successfully, you will end up at the Home page or the Reports page. If you have sample data installed, you may run the sample reports and browse the contents of the JapserServer Repository.

2.10 Schedule reporting configuration

The schdeuled reporting feature of JasperServer uses the open source Quartz scheduler (http://www.opensymphony.com/quartz). The following files in the WAR effect Quartz.

1. /WEB-INF/js.mail.properties

These settings are required in order to send email related to scheduled reports.

- Email server machine name

  Set this to be an SMTP email server reachable from the JasperServer machine. For example,

  ```
  report.scheduler.mail.sender.host=smtp.myorg.com
  ```

- Email user name and password

  A valid user name and password that can connect to the SMTP email server and send email. For example,

  ```
  report.scheduler.mail.sender.username=jsadmin
  report.scheduler.mail.sender.password=js.adminpassword
  ```

- "From" email address

  The email address that will appear as the "from" address on email sent from the scheduler. For example,

  ```
  report.scheduler.mail.sender.from=jsadmin@myorg.com
  ```

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

Properties passed into Quartz. The following are the JasperServer defaults.

- org.quartz.scheduler.instanceName=JasperServerScheduler
- org.quartz.threadPool.threadCount=2
- org.quartz.threadPool.threadPriority=3

You can tune more options by checking out the documentation at http://www.opensymphony.com/quartz/wikidocs/Configuration.html.

2.11 Problems with JasperServer Configuration

The most typical problems that users encounter are problems with database configuration. The following are some notes on checking the database configuration.

2.11.1 Database Configuration Settings

In the war file there are two files that have interaction with the database.These files are the following:

jasperserver/META-INF/context.xml   (main db config file)

Also:

<tomcat-install-dir>/conf/Catalina/localhost/JasperServer.xml

(This file will supersede the context.xml if it is used.)

(Tomcat will create this if/when Tomcat deploys JasperServer.)

And:

jasperserver/WEB-INF/hibernate.properties

jasperserver/WEB-INF/web.xml

2.11.2 Problems Running Reports

You may have your database configured properly but still get errors running reports.This is because (using the sample data) JasperServer allows the definition of datasources to be held in the JasperServer repository. These datasources can be used directly by reports that are also held inside the JasperServer repository. These resource definitions have their own settings, and the setting reference (and should match) the settings found in the configuration files mentioned in the section above.

If you are logged in as an "admin" user, you can browse the JasperServer Repository and view/edit the settings for the datasources that are used by the reports.

If you have the sample data, do a Repository Browse to /datasources. You should see two datasources:

JServerJdbcDS

JServerJNDIDS

You can edit these resources in order to view and/or update them. Check that the JDBC definition has the correct db username and password. You can double-check by logging into mysql using the same settings:

```
mysql -u <db-user> -p -h <hostname>     (will prompt for password)
```

Check that the JNDI definition points to a JNDI name space that is properly defined in your context.xml or jasperserer.xml file.

3 Web Services (SOAP) interface

This section describes the installation and configuration of the JasperServer Web Services application, iReport and the iReport Web Services plugin

The jasperserver.war is pre-configured to point to the following MySQL database setup:

Purpose:	JasperServer core data
url:	jdbc:mysql://localhost:3306/jasperserver
username:	root
password:	password

Purpose:	Sample application data
url:	jdbc:mysql://localhost:3306/sugarcrm
username:	root
password:	password

If you have installed JasperServer and you had chosen to create the sample database, then you will have the above databases on your system. However, if you do not have these databases then follow the steps in 2.5 to create them.

3.1 Check for Good Startup

You cannot "log into" the Web Services interface via a web browser, however, you can utilize a feature of our Apache AXIS SOAP implementation to see that the Web Services interface is configured and responding properly.

You can point your web browser to the following urls:

http://<your-host>:8080/jasperserver/services/repository?wsdl

This url will automatically generate the wsdl (Web Services Definition Language) that can be used by client applications to access Web Services.

If everything is set up correctly, you should see something like the following in your browser display:

(short clip of output)

```
<wsdl:definitions targetNamespace="http://org.apache.axis2/">
 <wsdl:types>
  <xs:schema targetNamespace="http://org.apache.axis2/xsd" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
   <xs:element name="loginRequest">
    <xs:complexType>
          <xs:sequence>
      <xs:element type="xs:anyType" name="element"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="loginResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element type="xs:anyType" name="return"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   etc....
```

The most typical problems that users encounter are problems with database configuration. See 2.11 for instructions on checking the database configuration.

4 Getting and Installing iReport and the iReport Plugin

The JasperServer Web Services Interface application now supports interaction with the popular iReport JasperReport creation tool.

There is a JasperServer plugin and an updated version of iReport available for download.

To get iReport go to:

http://ireport.sourceforge.net (follow download links)

Download one of the following:

- windows installer
- iReport tar or zip file
- iReport src zip (needs to be compiled)

To get the JasperServer iReport plugin go to:

http://jasperintel.sourceforge.net (follow download links)

Look for the file:

iReport-plugin-<ver>.zip

4.1 iReport and JasperServer Plugin Installation

4.1.1 Windows Installer

If you downloaded the Windows installer, start the installer and follow the instructions given by the installer wizard.

NOTE: Please take note of where the installation directory is. It is usually something like:

C:/Program Files/JasperSoft/iReport-<ver>

Or on a Windows install, it could be something like:

C:/Program Files/JasperSoft/iReport-<ver>/etc/skeletons/console-wrapperd

This special directory structure helps with the Windows menuing.

To start iReport:

Start Menu/All Programs/JasperSoft/iReport-<ver>/iReport-<ver>

4.1.2 Other Installations

If you downloaded the iReport-<ver>.zip or the iReport-<ver>.tar, you can unpack them into the directory of your choice. For instance:

Windows:

C:/iReport-1.3.1

Linux:

/opt/iReport-1.3.1

You can start iReport:

Windows:

<iReport-install-dir>/iReport.bat

Linux:

<iReport-install-dir>/iReport.sh

4.1.3 Get JasperServer Plugin

From the jasperintel.sourceforge.net site, get the iReport-plugin-<ver>.zip

http://jasperintel.sourceforge.net (follow download links)

4.1.4 Configure iReport and Plugin

Unzip the plugin into a new or existing directory.

What needs to be done is to copy the jar files and other plugin dependencies over to your iReport installation directory so that iReport will find everything it needs to run at application startup time.

Copy the following files:

1) JasperServerIRPlugin-<ver>.jar

   <plugin-dir>/JasperServerIRPlugin-<ver>.jar

   to:

   <iReport-install-dir>/lib

2) plugin.xml

   <plugin-dir>/src/ireport/plugin.xml

   to:

   <iReport-install-dir>/plugins

3) Plugin Dependency Jars

   <plugin-dir>/lib/*.*

   to:

   <iReport-install-dir>/lib

4.2 Run iReport with the Plugin

iReport should now have everything it needs to find and initialize the JasperServer plugin at startup time.

Start iReport (see Sections 4.1.1 and/or 4.1.2 above).

4.2.1 Point to Your JasperServer Web Service

If everything has been configured correctly, you will see a JasperServer pane in the upper left hand side of your iReport workspace.

If you click on the server icon you will get a dialog box titled "JasperServer Plugin" that allows for the JasperServer Access Configuration.

In this dialog box, you will enter the information that will enable you to connect to your JasperServer Web Service.

For instance:

Name:	MyServerName
JasperServer URL:	<you should only need to change the hostname>
Username:	<jasperadmin or tomcat>
Password:	<password or tomcat>

A server entry should now appear in this pane.

Double-click on the server entry to see a list of Resources at the root of the JasperServer repository. You may view the property information on the Resources in the repository. You may download, edit and upload jrxml files. You may download images and other files (using right-click "export file").

In addition, you may download, edit, and upload jrxml files to and from the server.

5 Configuring JasperServer for PostgreSQL

The instructions in Section 2 will configure JasperServer to use the MySQL database. This section describes the changes to that procedure required to support PostgresSQL. These instructions have been tested for PostgresSQL 8.1.6 on Windows XP and Tomcat 5.5.12 but should work on other versions of these packages.

5.1 Initialize database

Go to the scripts/postgresql directory under the JasperServer installation directory.

Start psql using an admin account (this could be postgres or something else):

```
psql –U postgres
```

At the psql prompt:

```
create database jasperserver encoding='utf8';

\c jasperserver

\i jasperserverCreate.ddl

\i jasperserverCreateDefaultSecurity.sql

\q
```

5.2 Create SugarCRM and Foodmart databases (optional)

Repeat the steps above to start psql.

At the psql prompt:

```
create database sugarcrm;

create database foodmart;

\c sugarcrm

\i sugarcrm.dump.sql

\c foodmart

\i foodmart.dump.sql
```

```
\q
```

5.3 Tomcat configuration

Copy a PostgreSQL JDBC driver JAR file to the `common/lib` directory in your Tomcat installation.

5.4 Updates for WAR

Use the procedure in section 2.1.6 to expand jasperserver.war manually into the Tomcat webapps/jasperserver directory. Once it is expanded, change the following files:

META-INF/context.xml

Set up the JNDI entries to point to your databases. Example entries are below (change user, password, host, and port as needed):

```
<Resource name="jdbc/jasperserver"

    auth="Container"

    type="javax.sql.DataSource"

    maxActive="100" maxIdle="30" maxWait="10000"

    username="postgres" password="postgres"

    driverClassName="org.postgresql.Driver"

    url="jdbc:postgresql://localhost:5432/jasperserver"

    defaultAutoCommit="false"/>

<Resource name="jdbc/sugarcrm"

    auth="Container"

    type="javax.sql.DataSource"

    maxActive="100" maxIdle="30" maxWait="10000"

    username="postgres" password="postgres"

    driverClassName="org.postgresql.Driver"

    url="jdbc:postgresql://localhost:5432/sugarcrm"/>

<Resource name="jdbc/foodmart"

    auth="Container"

    type="javax.sql.DataSource"

    maxActive="100" maxIdle="30" maxWait="10000"

    username="postgres" password="postgres"

    driverClassName="org.postgresql.Driver"

    url="jdbc:postgresql://localhost:5432/foodmart"/>
```

WEB-INF/js.quartz.properties

Add the following line (the backslash and line break are not necessary):

```
org.quartz.jobStore.driverDelegateClass=\

org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
```

WEB-INF/hibernate.properties

Change the Hibernate dialect setting:

```
metadata.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

At this point, start up Tomcat; you should be able to log in with `jasperadmin/newPassword`

5.5 Import sample repository data

Go to the scripts directory and change the following files:

ji-export-util/js.jdbc.properties

Change the Hibernate dialect setting and other JDBC settings:

```
metadata.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

metadata.jdbc.driverClassName=org.postgresql.Driver

metadata.jdbc.url=jdbc:postgresql://localhost:5432/jasperserver?defaultAutoCommit=fa
lse

metadata.jdbc.username=postgres

metadata.jdbc.password=postgres
```

ji-export-util/js.quartz.properties

Add the following line:

```
org.quartz.jobStore.driverDelegateClass=\

org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
```

Add the PostgreSQL JDBC jar to the ji-export-util/lib directory.

Now run the import script:

```
ji-import.bat --input-dir ji-catalog
```

The repository should now contain all the sample repository data.

Many of the sample reports use a datasource which refers to the sample SugarCRM database, but that datasource needs to be updated to point to your SugarCRM instance, which can be done with the following steps in JasperServer:

- Choose the Manage/Repository menu item
- Click on the datasources folder
- Click on the “edit” link for JserverJdbcDS
- Update the settings to match your SugarCRM database
- Click Save

The sample reports should now work with the data in your SugarCRM database.

5.6 Add triggers to clean up large objects (optional)

PostgreSQL handles BLOB (binary large object) data types differently from most databases. The actual BLOB data is stored in a separate table, called "pg_largeobject", and it is not automatically deleted when the row referring to it is deleted, so that the BLOB data will keep on taking up space unless a trigger is installed to delete the BLOB when the reference is updated or deleted.

JasperServer uses BLOB types in three types of repository objects:

- report content files (JIContentResource table)

- other file resources (JIFileResource table)
- compiled reports (JIRepositoryCache table)

The following PostgreSQL script will set up triggers on these three tables to delete large objects when their references are deleted:

```
create language 'plpgsql';
create or replace function lo_clean() returns trigger as '
declare
        lo_oid oid;
begin
        if (TG_OP = ''UPDATE'') then
                if (old.data = new.data) or (old.data is null) then
                        return new;
                end if;
        end if;
        select into lo_oid loid
          from pg_largeobject
          where loid = oid(old.data) and pageno=0;
        if found then
                perform lo_unlink(lo_oid);
        end if;
        return new;
end' language 'plpgsql';

create trigger ji_file_lo_cleanup
after delete or update on jifileresource
for each row execute procedure lo_clean();

create trigger ji_content_lo_cleanup
after delete or update on jicontentresource
for each row execute procedure lo_clean();

create trigger ji_cache_lo_cleanup
after delete or update on jirepositorycache
for each row execute procedure lo_clean();
```